Final Report on the "Software Product" Groupwork

A taxi management system

AlbaniaSuperUberApp Team: *Dalird Bufi, Melvir Singh, Xhulio Bishtaja*

1. Introduction

<u>Project Description</u>

This project is a school project for the Applied Software Engineering module. We have named the project 'AlbaniaSuperUberApp'. The motivation behind this product is 2 fold. Firstly, to be able to harness technology and be able to stamp our footprint on what we really want the current Uber / Taxi systems out there to be. By developing something from scratch, we are able to create features that we deem will be useful. Secondly, we would like to put into practice what we have learnt throughout the module. These include the programming technologies such as Flask and Python and also the project management aspects such as Scrum which we utilized to run the entire product from start to end.

The product was developed for a duration of about 1 month. Unfortunately, 1 month is never sufficient to deliver everything that we would like in an 'Uber' type of system but we focused on the core features and believe we have made it happen successfully via our 3 man development team.

We feel there are 3 aspects to the novelty of the product. Firstly, the product is based on a proper client/server based architecture. The benefit of this is that the server side can be started small (due to just being a 1 month development project work) with a handful of RESTful API services and be scaled up over time. Thus, the system can always be built-on and it will never be the end to it as the code base will remain manageable and sustainable. Secondly, we feel that by including all 3 user roles into the system, the product is self-sustaining. For example, we have the Customer, Driver and Provider roles. Each role comes with their own set of features and a user can simply sign up and be completely self-sustaining instead of requiring any form of manual intervention. Lastly, we have the Feedback feature in the product such that Customers and Drivers can provide feedback in real-time in the application that the Provider can take note of and potentially include in subsequent Sprints. We have to be upfront that there are no novel Customer features over the current Uber app in the market but it would be silly to expect such in a 1 month development timeframe. Moreover, in any proper product management lifecycle, core features should be prioritized and developed first prior to other unique/advanced/novel features.

Challenges

There were 2 main challenges we encountered throughout the project.

Firstly, other than Melvir, the other 2 members of the team are completely new to developing in the Client/Server architecture. Moreover, the architecture was covered at the very most 1 hour in a lesson and there was no practical lessons that the team members could try their hands on to try it out. Rather, this project became the practical lesson. Is that ideal? I think it depends on what the outcome and deliverable of the product is. However, we got through this challenge by Melvir sharing his experience in developing in such an architecture, giving concrete code examples as reference that the rest could follow. We also followed this up by spending lots of time and effort following online tutorials to learn the basics to be able to code in proper standards.

Secondly, it's also the first time that the 3 of us are working together on something. Naturally, there will be scenarios where we are learning/adapting to each other's working styles. Hence, we went through the typical group forming, storming, norming and performing lifecycle. We kept things objective on the project knowing that we had certain deliverables to meet. The scrum model did help as it helped to set tasks in smaller attainable ways.

Highlights

We are very pleased to report that we feel that the entire team has come to a good standard of being able to code well and produce features quickly in the Client/Server architecture by the end of the project. We also worked well in the Scrum model and all of us adapted to the Planning, Daily Scrum (over 3 days instead of daily) and Review meetings well and were very consistent throughout the project lifecycle. Furthermore, all 3 of us got to be good friends and will continue to network even after the end of this module.

2. Customer Needs

We feel that with some finetuning, the product could potentially be trialed in a proper market with Users able to sign up as either Customers or Drivers. Both these roles form the primary customers in the market for us. We do not have a specific planned customer in mind as we developed the product as an overall fit to the market and believe that it could be successful as a competitor to Uber with the features that we have built (and can potentially finetuned further).

We did User Journeys with some Customers and Drivers and realized that some of the pain-points they brought up (such as not being able to submit real-time feedback in app or not being able to export their Booking/Job history) which we have implemented in the product would definitely be useful and engaging to convince them to try out the product.

In terms of secondary stakeholders, we believe that they would include Citizens of Albania that we're launching the application in as well as the media in general. For the Citizens, it definitely allows them a different option to Speed Taxi App. The benefit of such is that should Uber start to raise prices unfairly, they would be able to compare the prices to our product and thus still have a viable option to continue booking taxis. For the media, it will be a 2 way opportunity for them and for us to be able to provide a review of the product. Should the product be successful, it might draw more views to their review and thus their media company overall as well.

The desired overall experience that we wish to accomplish for our stakeholders is a good, consistent, intuitive product for them to trial and use as an alternative to Speed Taxi App in Albania.

User Requirements

Customer User Stories:

- As a customer, I should be able to register, login, and log out so that I can access and manage my account.
  Acceptance Test: Given a registration page, when valid registration details are provided, then the account should be successfully created, and login should be possible.
- As a customer, I want to search for a ride by providing my current location so that I can find available ride options.
  Acceptance Test: Given the home screen, when I enter my locations, then the app should display available ride-hailing options.
- As a customer, I should be able to view my booking history so that I can keep track of my past rides.
  Acceptance Test: Given a booking history section, when I navigate to the history page, then I should see a list of my past bookings with relevant details.
- As a customer, I should be able to manage my payment methods so that I can easily make payments for my rides.
  Acceptance Test: Given the payment methods section, when I add, update, or delete a payment method, then the changes should be reflected in my account.
- As a customer, I should be able to provide feedback so that I can share my experience and contribute to service improvement.
  Acceptance Test: Given a feedback submission form, when I provide feedback, then it should be recorded for improvements.

Driver User Stories:

- As a driver, I should be able to register, login, and log out so that I can access and manage my driver account.
  Acceptance Test: Given a registration page, when valid registration details are provided, then the account should be successfully created, and login should be possible.
- As a driver, I want to view incoming booking requests so that I can accept and fulfill them.
  Acceptance Test: Given the driver app, when I access the booking requests page, then I should see a list of incoming requests.
- As a driver, I want to accept booking requests based on my availability and location so that I can efficiently manage my time and resources.
  Acceptance Test: Given a booking request, when I accept the request, then it should be assigned to me for fulfillment.
- As a driver, I should be able to view my booking job history so that I can track my past rides and earnings.
  Acceptance Test: Given a job history section, when I navigate to the history page, then I should see a list of past jobs with relevant details.
- As a driver, I should be able to manage my car details so that I can keep them up to date.
  Acceptance Test: Given the car details section, when I add, update, or delete car details, then the changes should be reflected in my profile.

Provider User Stories:

- As a provider, I should be able to register, login, and log out so that I can access and manage my provider account.
  Acceptance Test: Given a registration page, when valid registration details are provided, then the account should be successfully created, and login should be possible.

- As a provider, I should be able to track car positions and reset them for testing purposes so that I can monitor and manage the fleet.
  Acceptance Test: Given the provider admin panel, when I access the car positions section, then I should see real-time car locations and be able to reset them.
- As a provider, I want to review customer feedback so that I can assess service quality and make improvements.
  Acceptance Test: Given the feedback section, when I access the feedback list, then I should see all feedback provided by customers.

Documentation

Personas:

- Ben is a driven and extroverted infrastructure engineer at Apple, passionate about solving real-world problems.
- John is a working adult in the banking sector, relying on transportation to reach work on time for daily scrum meetings.
- May is a customer service officer in an MNC, expecting quick and reliable transport options when public transport is unavailable.
- Robert is an experienced taxi driver, focused on maximizing earnings during his daily shifts.
- Theresa is a software developer in Google, passionate about developing robust and performant applications.

Scenarios:

- Customer Scenario: Booking a ride to a specific car location.
- Driver Scenario: Accepting and completing a ride request from a specific car location.
- Provider Scenarios: Managing vehicle positions and reviewing feedback.

User Stories:

- User stories for Customers include scenarios such as registration, login, booking a ride, managing payment methods, and providing feedback.
- User stories for Drivers include registration, login, accepting booking requests, managing cars, and providing feedback.
- User stories for Providers include registration, login, managing cars and feedback.

Feature Identification:

The feature identification includes features for registration and login, booking rides, managing payment methods, viewing booking history, providing and managing feedback, accepting booking requests, managing cars, and reviewing feedback.

3. Project Goals

Since Albania does not have an Uber like app, we wanted to deliver a product that could be a solid competitor to the taxis and their current Speed Taxi app there.

Thus, the main customer problem we chose to solve and that we wanted to improve the experience of was to ensure the entire booking process intuitive and smooth. For the customer, we also wanted them to be able to provide us and the providers real-time feedback.

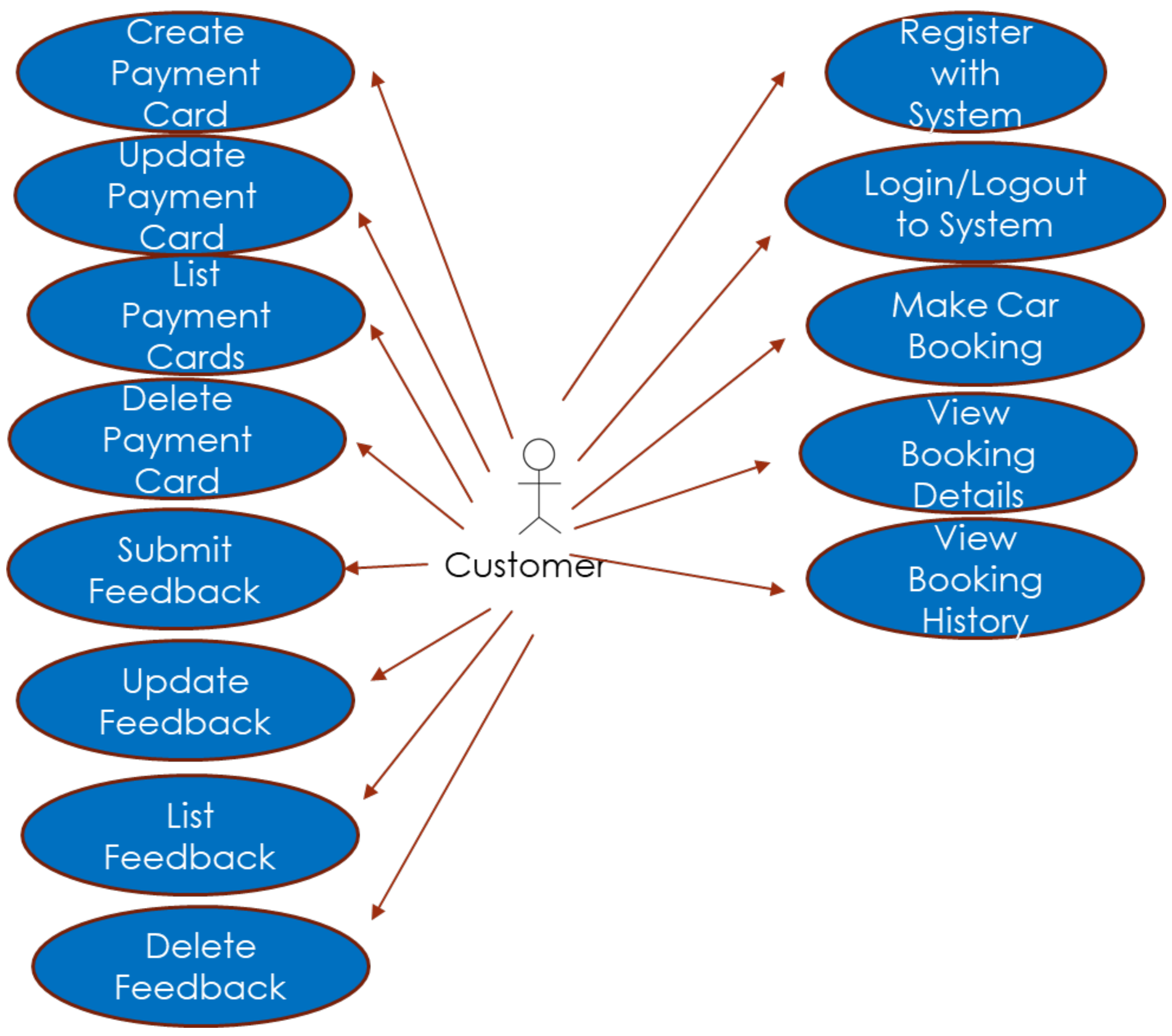
The main driver problem we wanted to solve was to allow them a comprehensive overview of all the jobs they have undertaken. Compared to the Speed Taxi app, the drivers feedback that the interface was buggy and often resulted in errors leading to them not being able to view their job history.

For the providers, the main problem we wanted to solve was that there was no current way to have a Command & Control view of all their cars in real-time. Thus, we wanted to achieve a C&C view for them to have a quick overview and sensing of all the cars and their respective locations.

We believe that by focusing on these 3 issues, we would be able to deliver a product that fit well for all 3 roles that we are releasing the product for – Customers, Drivers, Providers. The benefits would not only be in terms of time saved but also in terms of effort saved. We validated these ideas during the User Journeys with our users (covered above) and they were all very positive about the experience and benefits that our product will come with. That was indeed very encouraging.

Uses Cases

For Customer Role

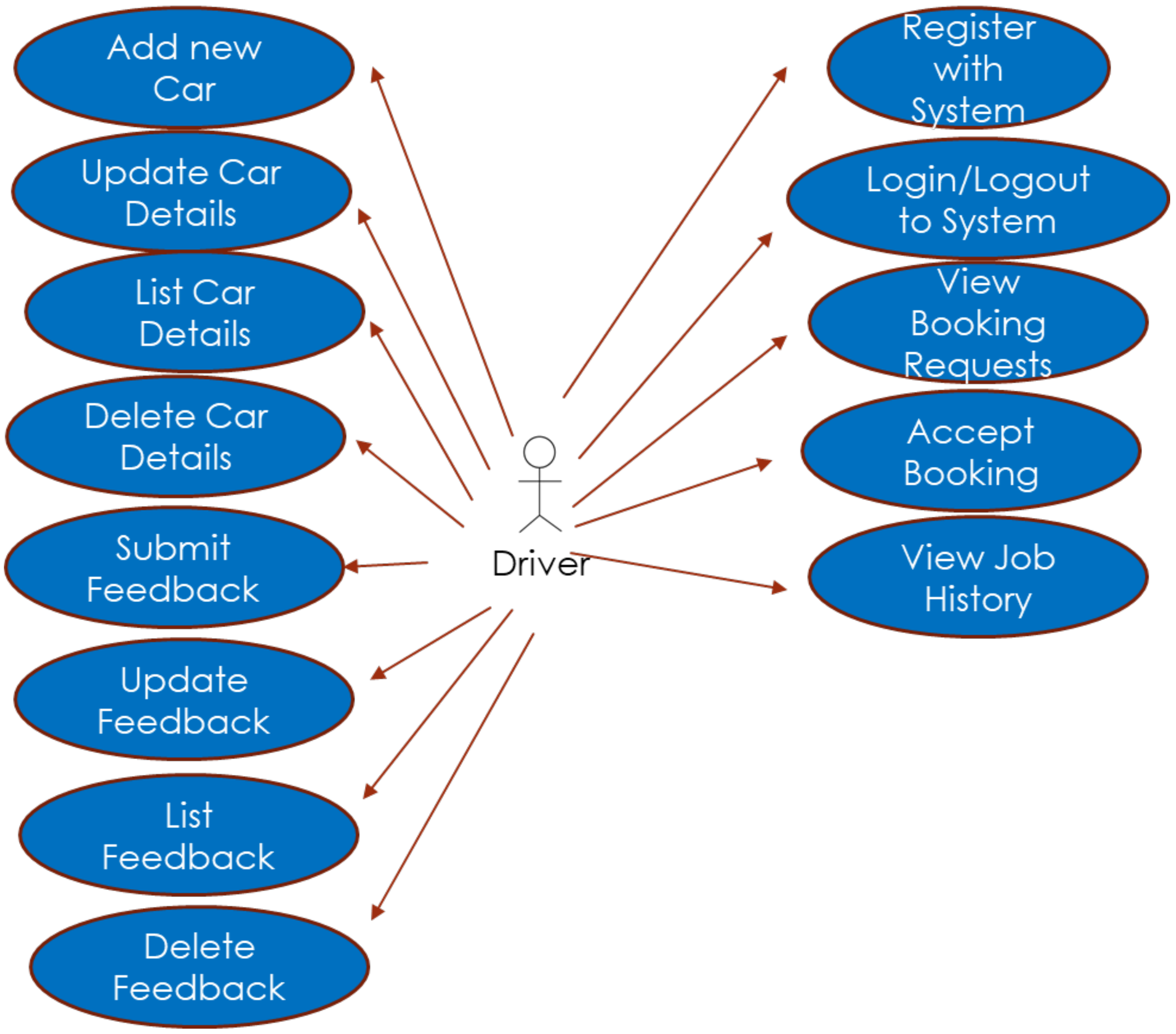


Detailed Description of 1 use-case for Customer Role

Use-Case	Make Car Booking	
Description	Customer shall be able to make a booking for a Car to get from Current Location to Destination	
Precondition	The user has registered successfully with 'Customer' role and has also logged in successfully.	
Sequence	Step	Action
	1	User logs into application

	2	User clicks on ‘Book Car’ button on top navigation bar
	3	User inputs in Current Location (eg. Wembley) and Destination (eg. Liverpool)
	4	User clicks on ‘Book’ button
	5	System updates user that the request has been sent and awaiting for acceptance by any Driver
	6	User clicks on ‘Check if accepted by Driver (again)’. Assumption that a Driver has already accepted
	7	System updates user with Car details
	8	User clicks on ‘Simulate Starting of Ride’
	9	System updates user that Ride is started and updates Driver’s car pos X and pos Y with the Current Location specified
	10	User clicks on ‘Simulate Completion of Ride’
	11	System updates user that Ride is completed and updates Driver’s car pos X and pos Y with the Destination specified
	12	User selects Driver Rating and clicks on ‘Rate your Driver’
	13	System thanks user for the rating and informs the end of flow
Postcondition	Driver’s car’s pos X and pos Y are updated to the Destination longitude and latitude	
Comments	The steps are longer than a typical use-case on Uber as the current application is not able to automatically trigger starting and completion of ride and thus requires the User to push some buttons through the process	

For Driver Role

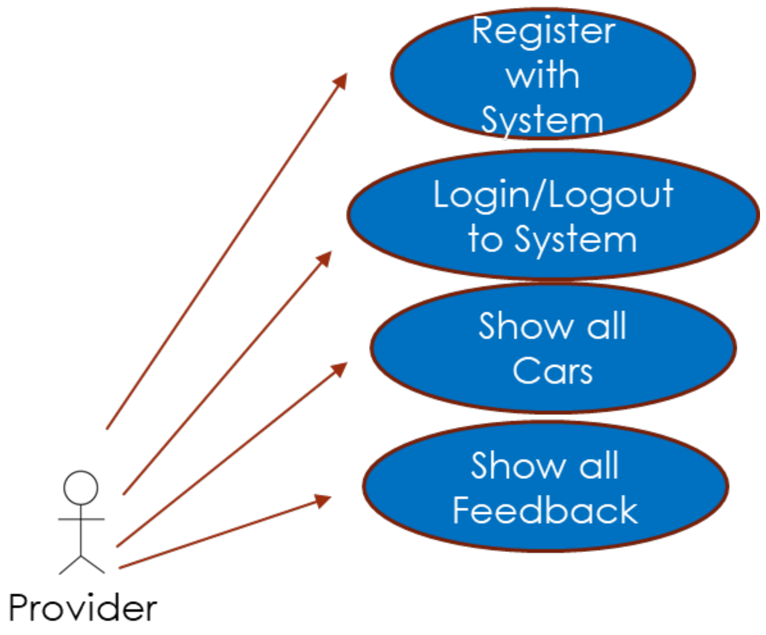


Detailed Description of 1 use-case for Driver Role

Use-Case	Accept Booking
Description	Driver shall be able to accept booking requests from Customers to get them from Current Location to Destination
Precondition	The user has registered successfully with ‘Driver role and has also logged in successfully.

Sequence	Step	Action
	1	User logs into application
	2	User clicks on ‘View Booking Requests’ button on top navigation bar
	3	User looks through the Booking Requests that appear in a table format
	4	User clicks on ‘Accept’ button for the ride that he/she wishes to accept
	5	System updates user that the request has been accepted and that the Customer will be notified
	6	System updates Booking table with the row’s ID to ‘Accepted by Driver’ status and car_id to ID of Driver’s car
Postcondition	The accepted booking request’s row will be updated in the database with ‘Accepted by Driver’ status and the car_id will also be populated	
Comments	The steps are straightforward and intentionally allows the Driver to freely select which job he/she wishes to accept rather than only showing those requests which are very close to him	

For Provider Role



Detailed Description of 1 use-case for Driver Role

Use-Case	Show all Cars	
Description	Provider shall be able to show all the cars in their system into a nice overview of Map to have a Command & Control sort of view	
Precondition	The user has registered successfully with ‘Provider’ role and has also logged in successfully.	
Sequence	Step	Action
	1	User logs into application
	2	User clicks on ‘Show all Cars’ button on top navigation bar
	3	Map loads with all the Cars in their correct positions
	4	User can click on any of the Cars that appears on the map to see the car brand, model and ids
Postcondition	Page loads without any errors	

Comments	The steps are straightforward and allows the Provider to see an overview of all cars on the map

Measures of Success

We believe that measuring success of our product and its use-cases are critical. Thus, we plan on doing this via User Interviews and also Surveys. For the former, we plan on meeting up with the same users that we did the User Interviews with to check in with them after they have trialed the app. This will allow them to provide an honest assessment on whether their pain points discussed before had indeed been addressed by our product. For the latter. We plan on coming up with a Survey questionnaire asking users on aspects such as benefits, functionalities, intuitiveness, UI/UX. We would then compile the results and assess how successful our product has been. We would also be able to prioritize their feedback by creating PBIs to add on to our Product Backlog.

4. System Description

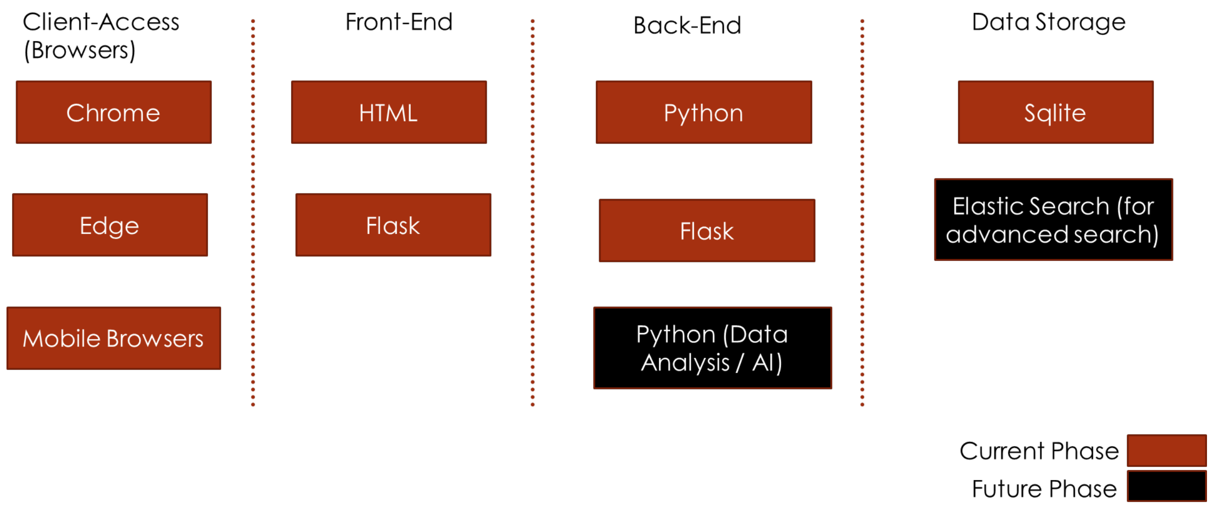
System Overview

The product uses the Client/Server architecture, thus having separate frontend and backend applications. There are various benefits to doing a product in such an architecture. Firstly, it allows newcomers to be able to understand each part easier and quicker as the codebase is naturally smaller due to it being split. Secondly, deployment is also more straightforward as any changes in frontend will only require re-deployment of the frontend without affecting the backend. Lastly, there are proper segregated concerns of responsibilities for having them separately. For example, the frontend is purely responsible for handling client (user) requests before sending the payloads to the backend. The backend is purely responsible for business logic and to perform data manipulation to/from the database.

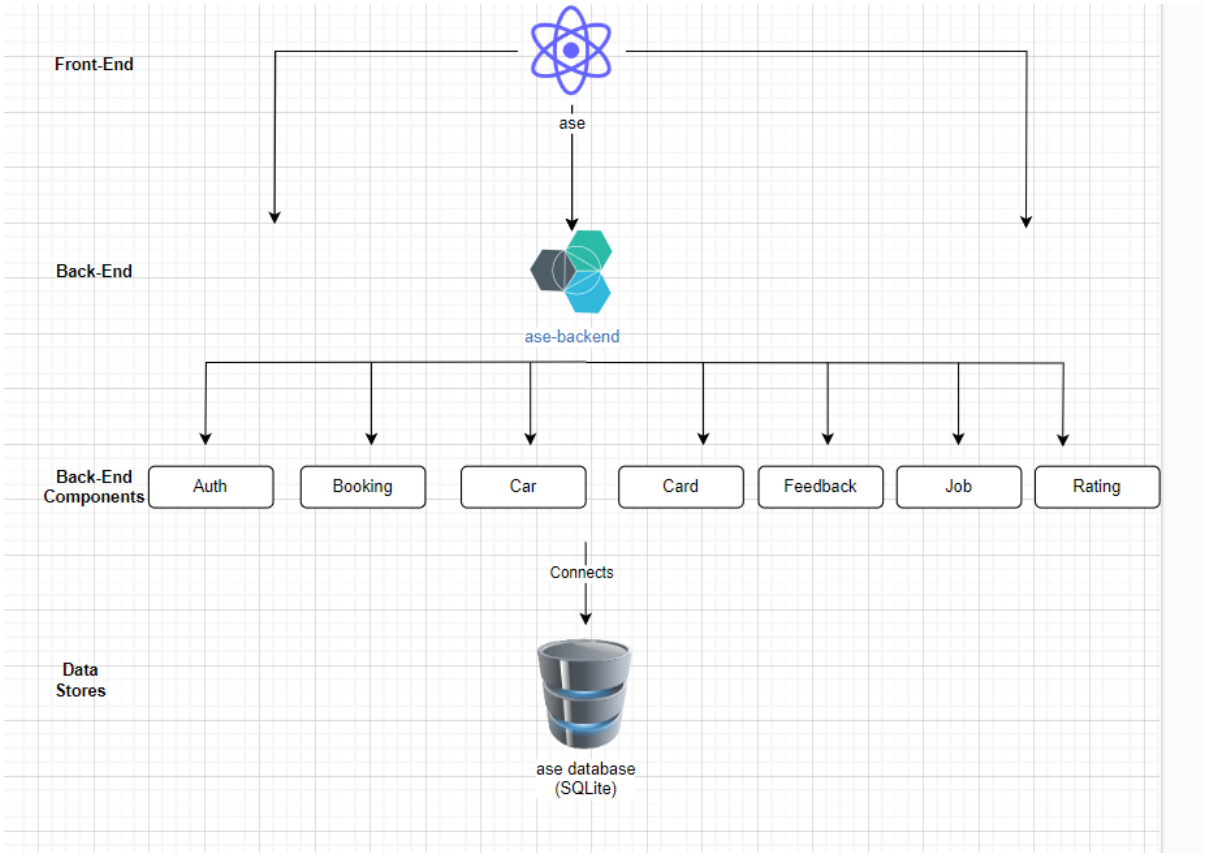
Naturally, there are challenges to this architecture too. Firstly, it is definitely more complex and complicated to code in compared to having both frontend/backend together in a single application. The technologies are also different (one contains flask/html whilst the other contains flask/python) and thus developers would need a greater range of skillsets. Secondly, it can also be harder to deploy as the team needs to ensure both deployments are running properly compared to just having to monitor one if both were packaged together.

However, the team feels that the benefits greatly outweighs the drawbacks thus we decided to proceed with this architecture. Should the team pick up more knowledge on tech standards in future the backend could be further decoupled/decomposed into several different backends to take advantage of the benefits that microservices entail as well.

Technology Stack



Application Architecture



There is a single frontend application that interacts with a single backend application. However, the backend application serves API routes on various different components. Each of this components take care of their own set domain boundary. For example, all authentication (such as register/login/logout) APIs are stored in the auth.py file. It makes it self-explanatory and also aids troubleshooting allowing faults to be narrowed much quicker. If there are any new authentication data required from the database, a new API route can be created in the auth.py file which will then be served by the ase-backend service.

Below is a summary of the API routes created for each backend component and also the team member that was in-charge to lead the development for it. Naturally, the same person who worked on that backend component also worked on the same frontend component to take care of the display on the pages. However, in light of having consistent code, we came up with proper code standards and flow. Then, with that as reference code and design flow, all of us managed to code well with proper standards. Thus, a new developer joining would not get a shock encountering completely different code and flow from page to page.

<u>Components Breakdown & Member in-charge</u>

Back-end Component	API routes created	Description of their usage	Team Member in-charge
Auth	/api/register /api/login /api/<int:id>/loadUser	For user registration For user login To load user information into session upon successful login	Melvir

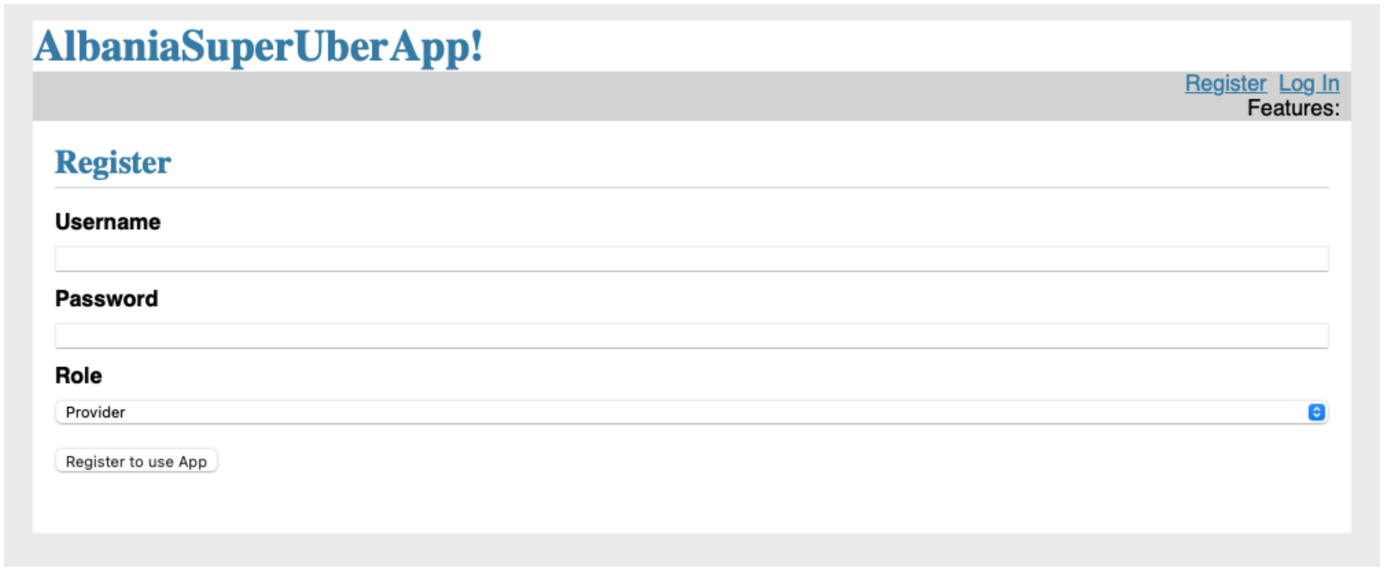
Booking	/api/showCars /api/bookcar /api/listRequests /api/checkBooking /api/<int:id>/acceptJob /api/<int:id>/getCarId /api/<int:id>/getCarDetails /api/listBookings /api/startBooking /api/completeBooking /api/rateDriver	To display cars for Provider To start car booking process for Customer To list all requests for Driver to accept To check if booking has been accepted by any Driver For Driver to accept job To retrieve car ID To retrieve car details For Driver to show jobs history To simulate ride start once Driver arrived at location To simulate ride completion once Driver arrived at destination To rate driver	Dalird
Car	/api/listCarDetails /api/createCar /api/<int:id>/getCar /api/<int:id>/updateCar /api/<int:id>/deleteCar	To list car details To add new car for new Driver To get car from car ID To update car details To delete car for Driver	Xhulio
Card	/api/listCard /api/createCard /api/<int:id>/getCard /api/<int:id>/updateCard /api/<int:id>/deleteCard	To list payment card details To add new payment card for Customer To get payment card from card ID To update payment card To delete payment card for Customer	Melvir

Feedback	/api/listFeedback /api/listAllFeedback /api/createFeedback /api/<int:id>/getFeedback /api/<int:id>/updateFeedback /api/<int:id>/deleteFeedback	To list all feedback details To list all Feedback for Providers To create new feedback To get feedback from feedback ID To update feedback To delete feedback	Melvir
Job	/api/listJob	To list all booking job history for Drivers	Dalird
Rating	/api/driverRating	To get rating for Drivers	Xhulio

5. Final Status

- Application (with screenshots)
- Unit Testing
- Line of Codes
- Codebases

Application (with screenshots):


AlbaniaSuperUberApp!

Register

Username

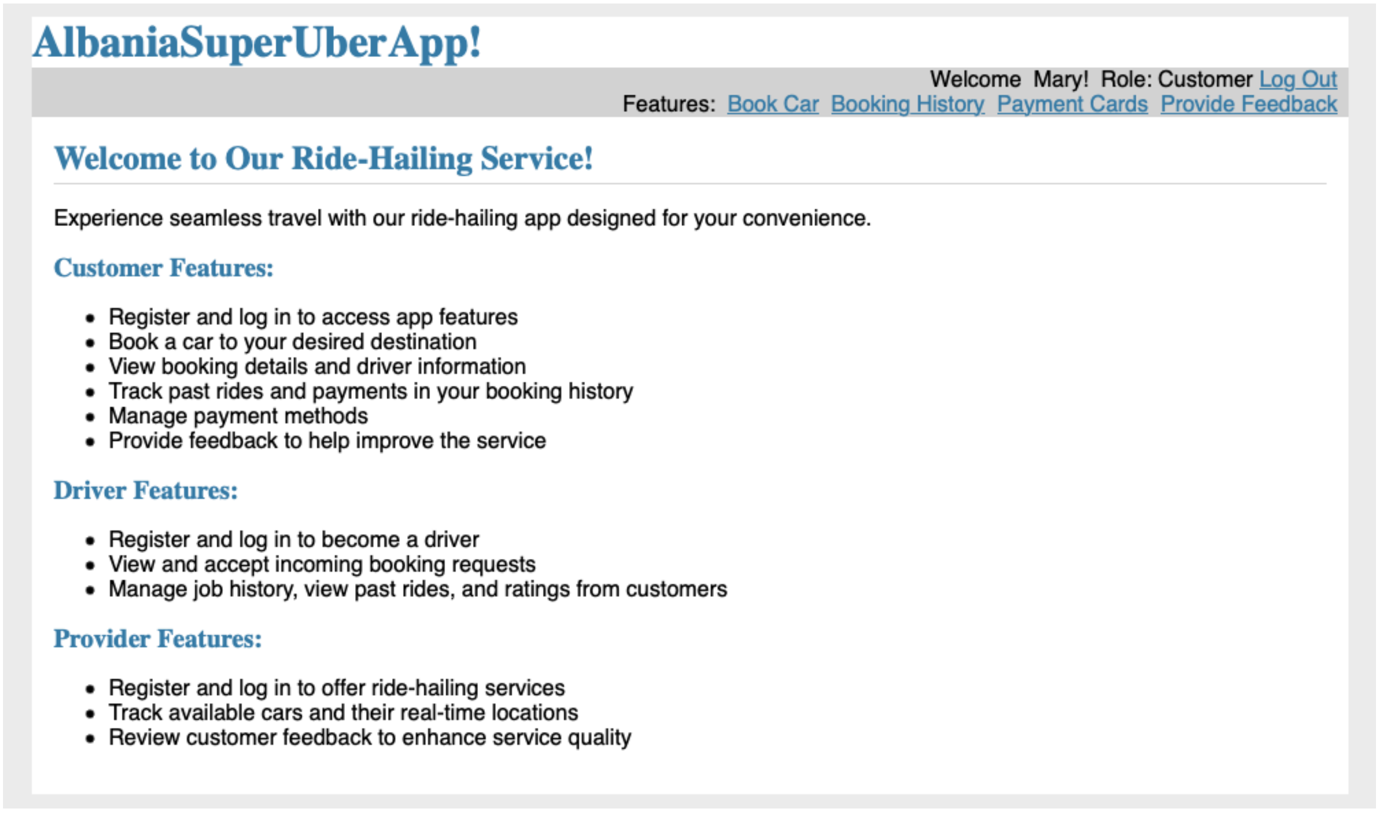
Password

Role

Provider

Figure 1. Screenshot showing where the user can to register a new account

Description: The user is presented with a registration form to create a new account. They are required to enter their username, password and the role. Upon filling in the required information, they can proceed to the next step.



AlbaniaSuperUberApp!

Welcome Mary! Role: Customer Log Out

Features: Book Car Booking History Payment Cards Provide Feedback

Welcome to Our Ride-Hailing Service!

Experience seamless travel with our ride-hailing app designed for your convenience.

Customer Features:

- Register and log in to access app features
- Book a car to your desired destination
- View booking details and driver information
- Track past rides and payments in your booking history
- Manage payment methods
- Provide feedback to help improve the service

Driver Features:

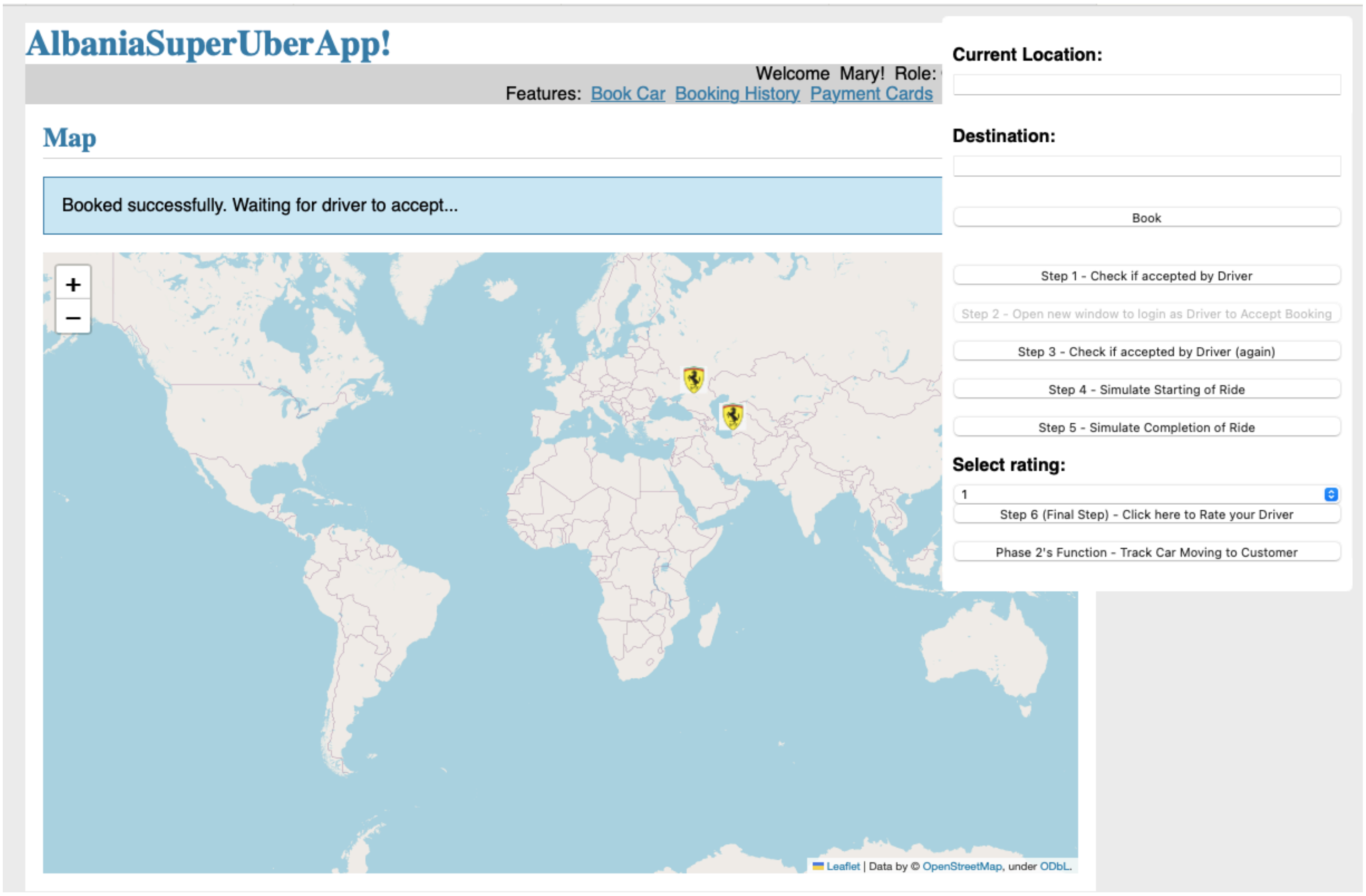
- Register and log in to become a driver
- View and accept incoming booking requests
- Manage job history, view past rides, and ratings from customers

Provider Features:

- Register and log in to offer ride-hailing services
- Track available cars and their real-time locations
- Review customer feedback to enhance service quality

Figure 2. Screenshot showing the default home screen and the user, successully logs in

Description: Default home screen with the announcements and all the customer features after the successfully log in.


AlbaniaSuperUberApp!

Welcome Mary! Role:
Features: Book Car Booking History Payment Cards

Map

Booked successfully. Waiting for driver to accept...

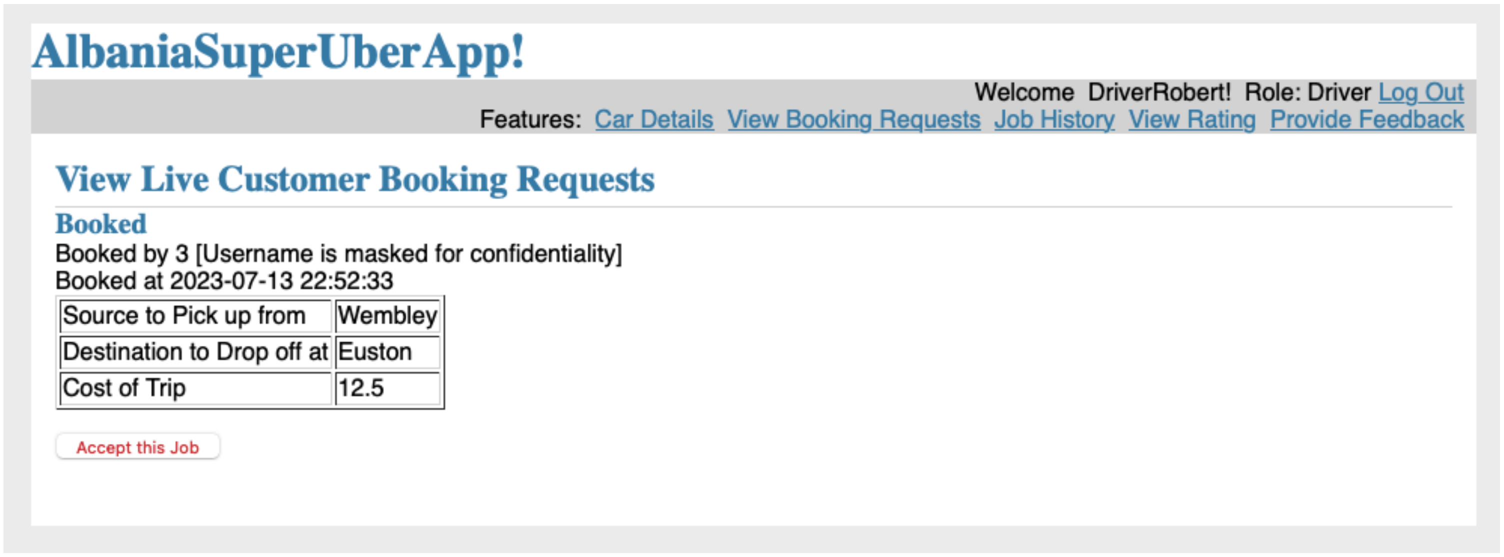
Current Location:

Destination:

Book
Step 1 - Check if accepted by Driver
Step 2 - Open new window to login as Driver to Accept Booking
Step 3 - Check if accepted by Driver (again)
Step 4 - Simulate Starting of Ride
Step 5 - Simulate Completion of Ride
Select rating:
1
Step 6 (Final Step) - Click here to Rate your Driver
Phase 2's Function - Track Car Moving to Customer
Leaflet | Data by © OpenStreetMap, under ODbL.

Figure 3. Screenshot showing the successfully booking by the customer

Description: After successfully booking a ride, the customer is shown a confirmation screen indicating that their booking has been successfully made. The screen informs the customer that they are now in the process of waiting for a driver to accept their request.


AlbaniaSuperUberApp!

Welcome DriverRobert! Role: Driver Log Out

Features: Car Details View Booking Requests Job History View Rating Provide Feedback

View Live Customer Booking Requests

Booked

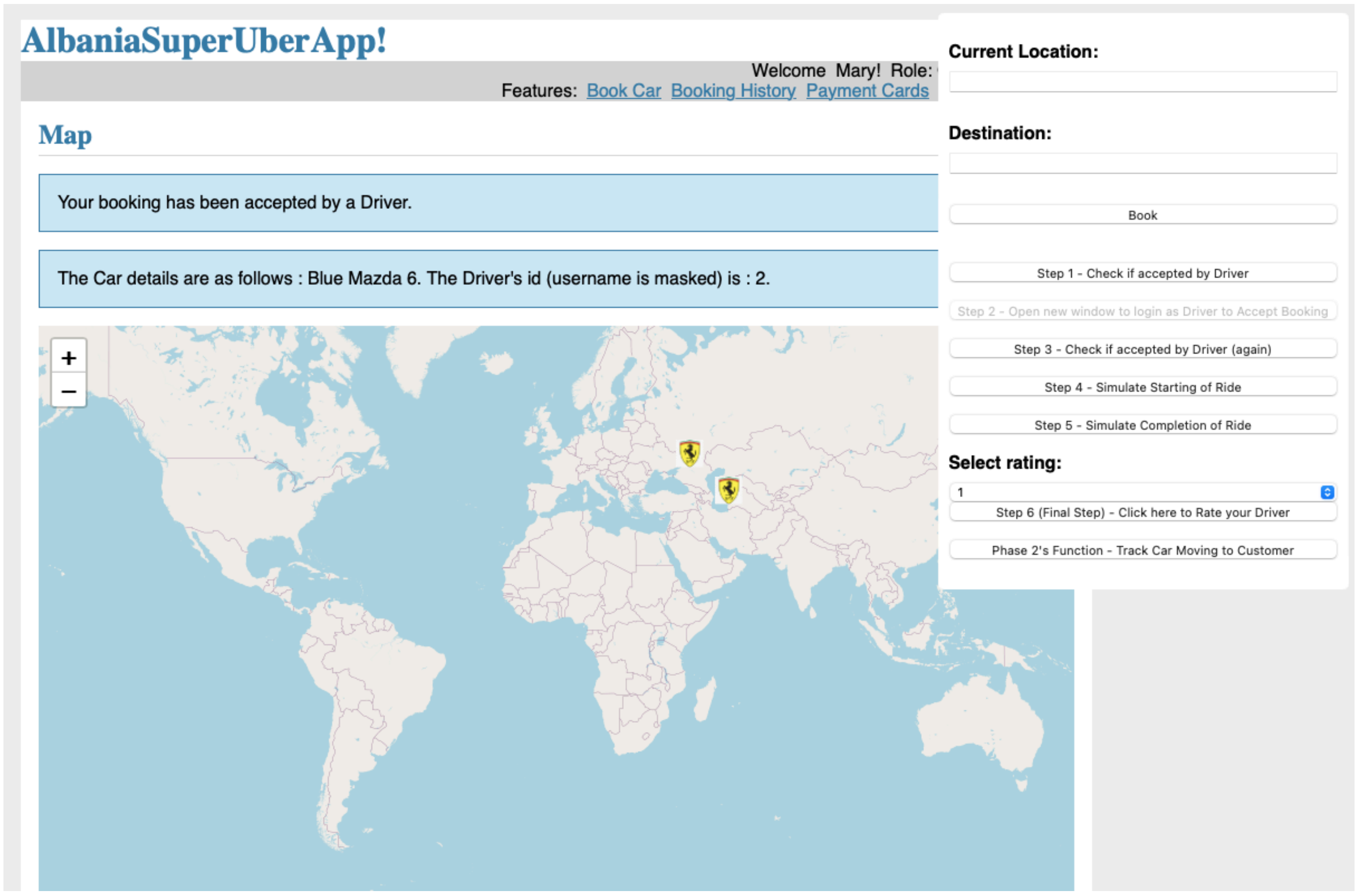
Booked by 3 [Username is masked for confidentiality]

Booked at 2023-07-13 22:52:33

Source to Pick up from	Wembley
Destination to Drop off at	Euston
Cost of Trip	12.5

Figure 4. Screenshot showing the view booking requests for the driver

Description: The View Booking Requests feature provides drivers with a comprehensive list of incoming ride requests. In this screenshot, we can see a specific request from Wembley to Euston and the cost of trip.

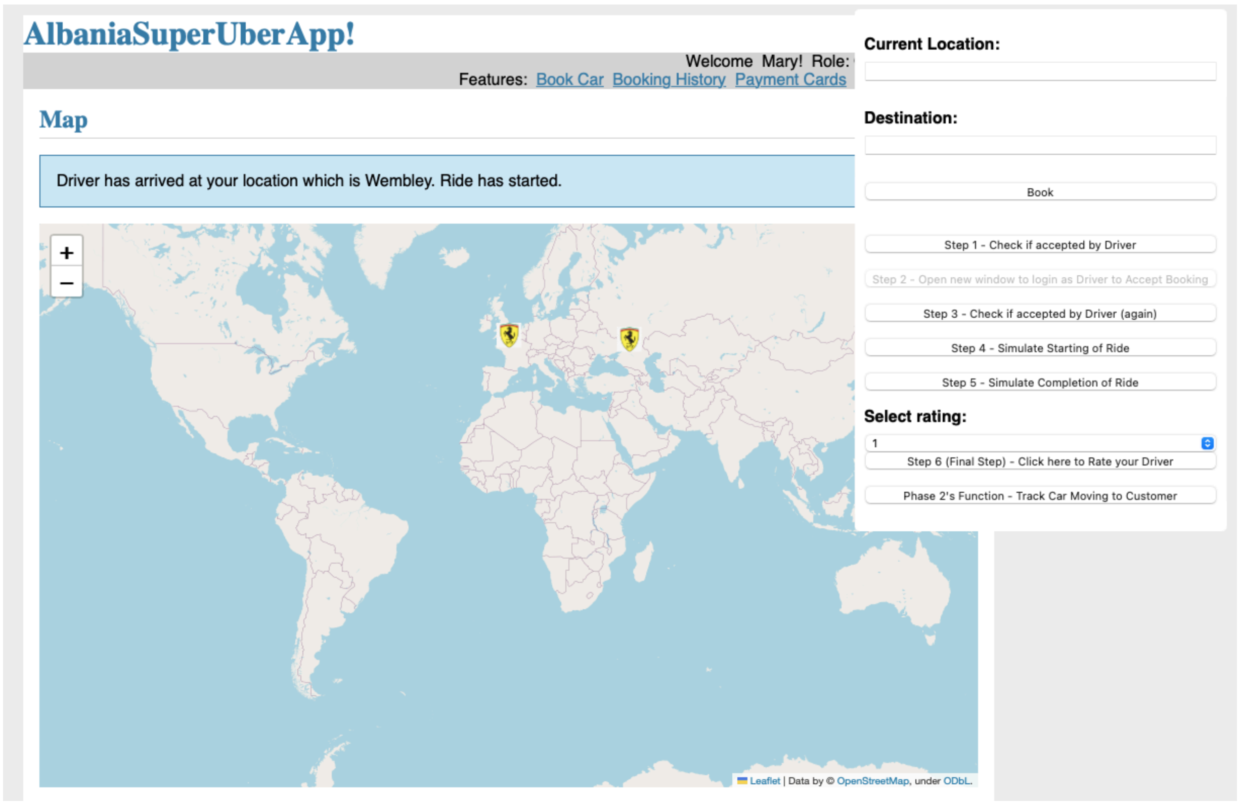

AlbaniaSuperUberApp!
Welcome Mary! Role:
Features: Book Car Booking History Payment Cards
Map
Current Location:
Destination:
Book
Step 1 - Check if accepted by Driver
Step 2 - Open new window to login as Driver to Accept Booking
Step 3 - Check if accepted by Driver (again)
Step 4 - Simulate Starting of Ride
Step 5 - Simulate Completion of Ride
Select rating:
1
Step 6 (Final Step) - Click here to Rate your Driver
Phase 2's Function - Track Car Moving to Customer

Your booking has been accepted by a Driver.

The Car details are as follows : Blue Mazda 6. The Driver's id (username is masked) is : 2.

Figure 5. Screenshot showing that the booking has been accepted by a driver

Description: Upon acceptance of the booking request by a driver, the customer is presented with a screen confirming that a driver has been assigned to their ride. This screenshot displays the driver's id and the car details.


AlbaniaSuperUberApp!

Welcome Mary! Role:

Features: Book Car Booking History Payment Cards

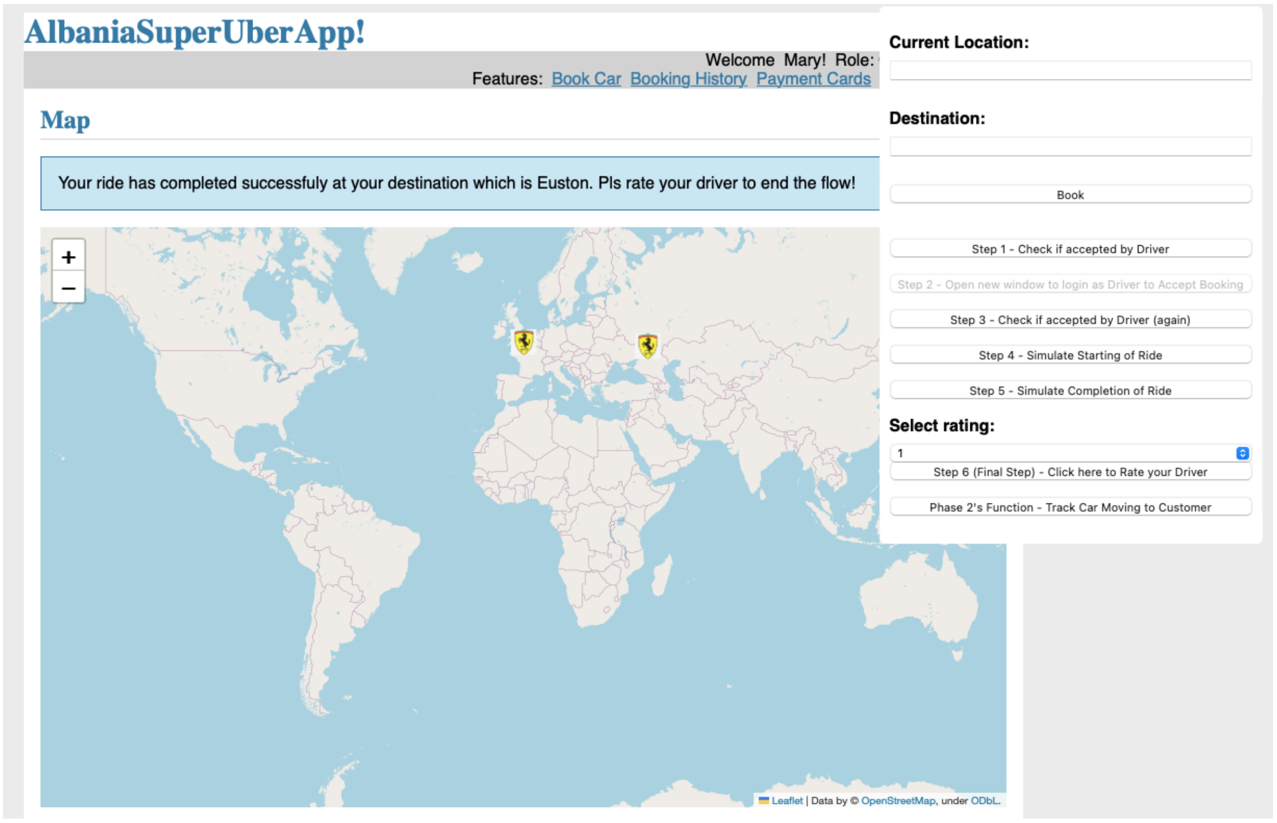
Map

Driver has arrived at your location which is Wembley. Ride has started.

Current Location:
Destination:
Book
Step 1 - Check if accepted by Driver
Step 2 - Open new window to login as Driver to Accept Booking
Step 3 - Check if accepted by Driver (again)
Step 4 - Simulate Starting of Ride
Step 5 - Simulate Completion of Ride
Select rating:
1
Step 6 (Final Step) - Click here to Rate your Driver
Phase 2's Function - Track Car Moving to Customer
Leaflet | Data by © OpenStreetMap, under ODbL.

Figure 6. Screenshot showing that the driver has arrived and the ride has started

Description: Once the driver arrives at the pickup location, the customer is presented with a screen confirming that the driver has arrived and the ride has started.


AlbaniaSuperUberApp!

Welcome Mary! Role:

Features: Book Car Booking History Payment Cards

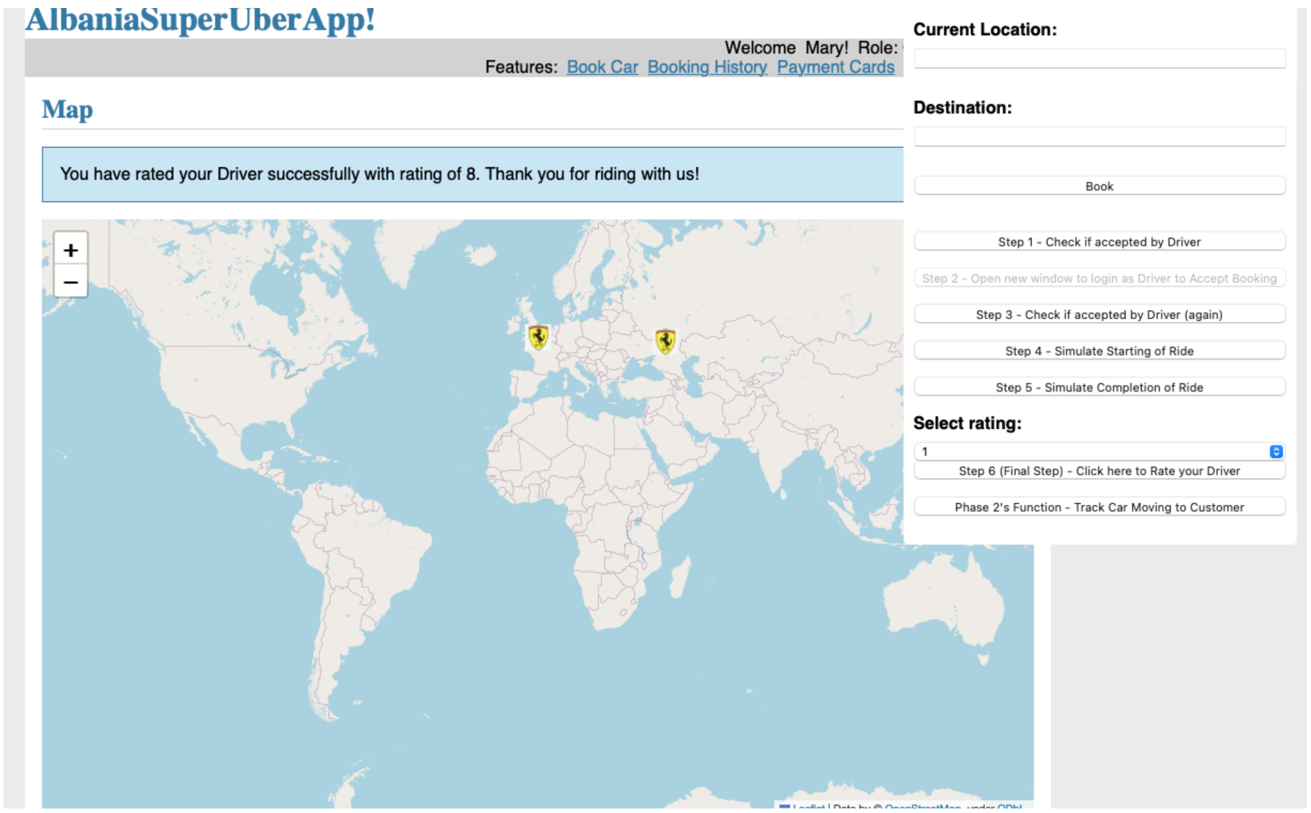
Map

Your ride has completed successfuly at your destination which is Euston. Pls rate your driver to end the flow!

Current Location:
Destination:
Book
Step 1 - Check if accepted by Driver
Step 2 - Open new window to login as Driver to Accept Booking
Step 3 - Check if accepted by Driver (again)
Step 4 - Simulate Starting of Ride
Step 5 - Simulate Completion of Ride
Select rating:
1
Step 6 (Final Step) - Click here to Rate your Driver
Phase 2's Function - Track Car Moving to Customer
Leaflet | Data by © OpenStreetMap, under ODbL.

Figure 7. Screenshot showing that the ride is been completed

Description: This screenshot shows that the ride has been successfully concluded, and the customer is now prompted to give a rating to the driver.


AlbaniaSuperUberApp!

Welcome Mary! Role:

Features: Book Car Booking History Payment Cards

Map

You have rated your Driver successfully with rating of 8. Thank you for riding with us!

Current Location:

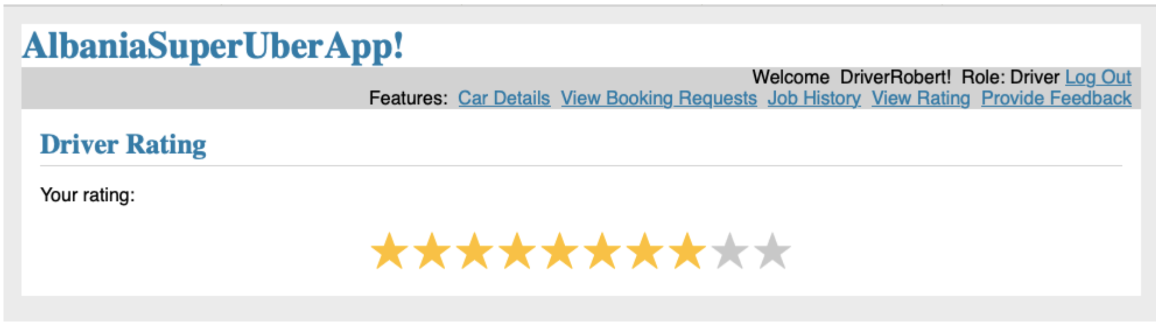
Destination:

Select rating: 1

Book
Step 1 - Check if accepted by Driver
Step 2 - Open new window to login as Driver to Accept Booking
Step 3 - Check if accepted by Driver (again)
Step 4 - Simulate Starting of Ride
Step 5 - Simulate Completion of Ride
Step 6 (Final Step) - Click here to Rate your Driver
Phase 2's Function - Track Car Moving to Customer

Figure 8. Screenshot showing that the rating has been successfully

Description: After providing a rating for the driver, the customer is presented with a screen confirming that the rating has been successfully submitted. This screenshot showcases the rating given to the driver, which in this case is 8/10.


AlbaniaSuperUberApp!

Welcome DriverRobert! Role: Driver Log Out

Features: Car Details View Booking Requests Job History View Rating Provide Feedback

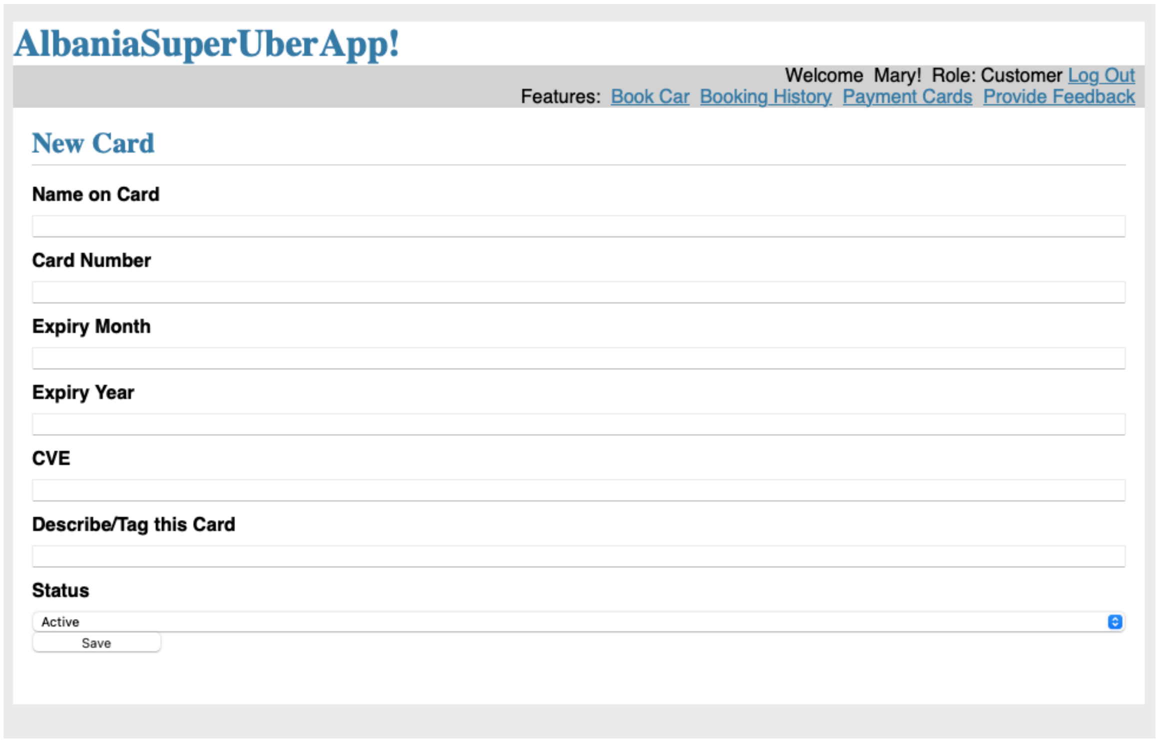
Driver Rating

Your rating:

★★★★★★★★☆☆

Figure 9. Screenshot showing the driver rating

Description: This screenshot showcases the driver's rating displayed as stars, with a rating of 8 out of 10. In this example, 8 stars are filled in yellow, representing the positive rating received by the driver, while 2 stars are left empty.


AlbaniaSuperUberApp!

Welcome Mary! Role: Customer Log Out

Features: Book Car Booking History Payment Cards Provide Feedback

New Card

Name on Card

Card Number

Expiry Month

Expiry Year

CVE

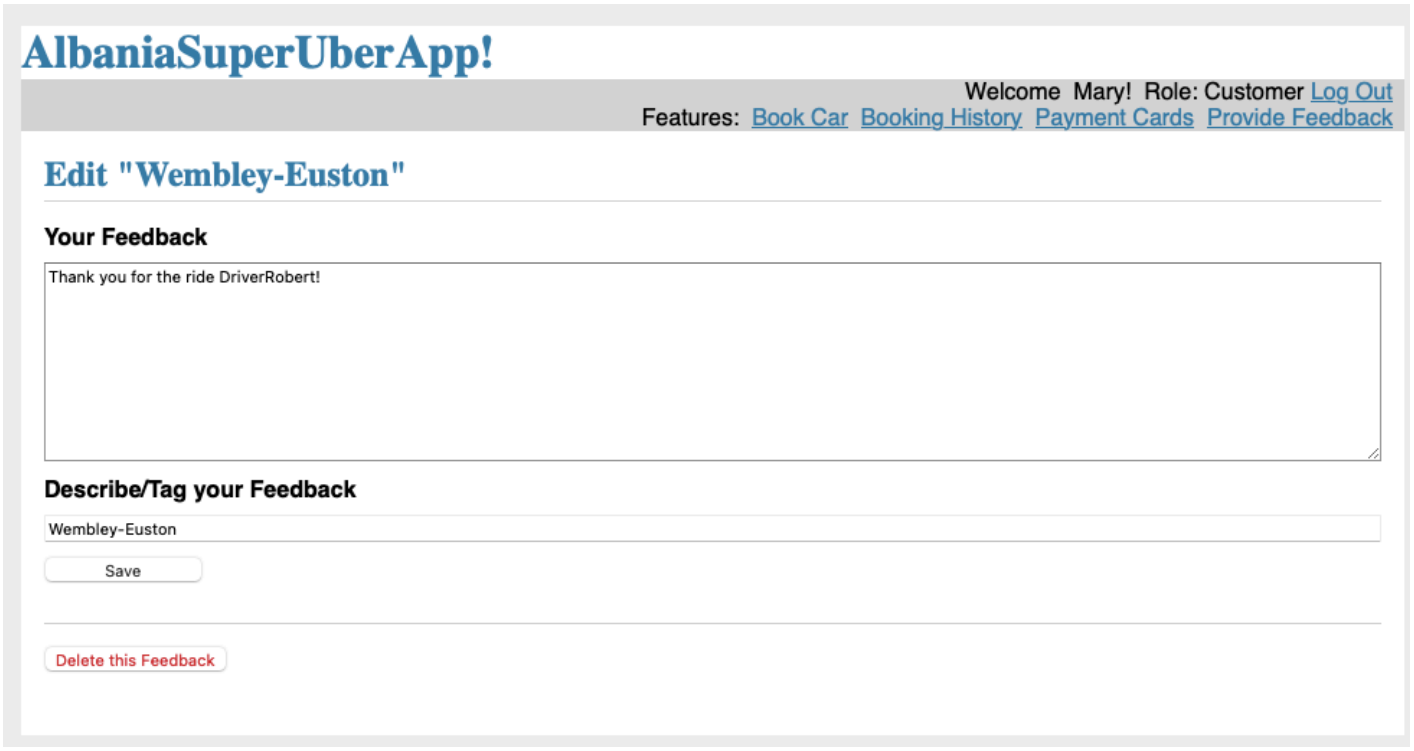
Describe/Tag this Card

Status

Active

Figure 10. Screenshot showing the payment feature for the customer

Description: This screenshot illustrates the payment feature for the customer, providing an overview of their saved cards. It shows the card details, including the card number, expiration date, name, and status. The screen allows customers to manage their payment options conveniently. After saving a card, customers have the ability to edit the card details, such as updating the expiration date or the status. They can also choose to delete a saved card if it is no longer needed. This flexibility ensures that customers can easily manage and maintain their payment methods within the app.


AlbaniaSuperUberApp!

Welcome Mary! Role: Customer Log Out

Features: Book Car Booking History Payment Cards Provide Feedback

Edit "Wembley-Euston"

Your Feedback

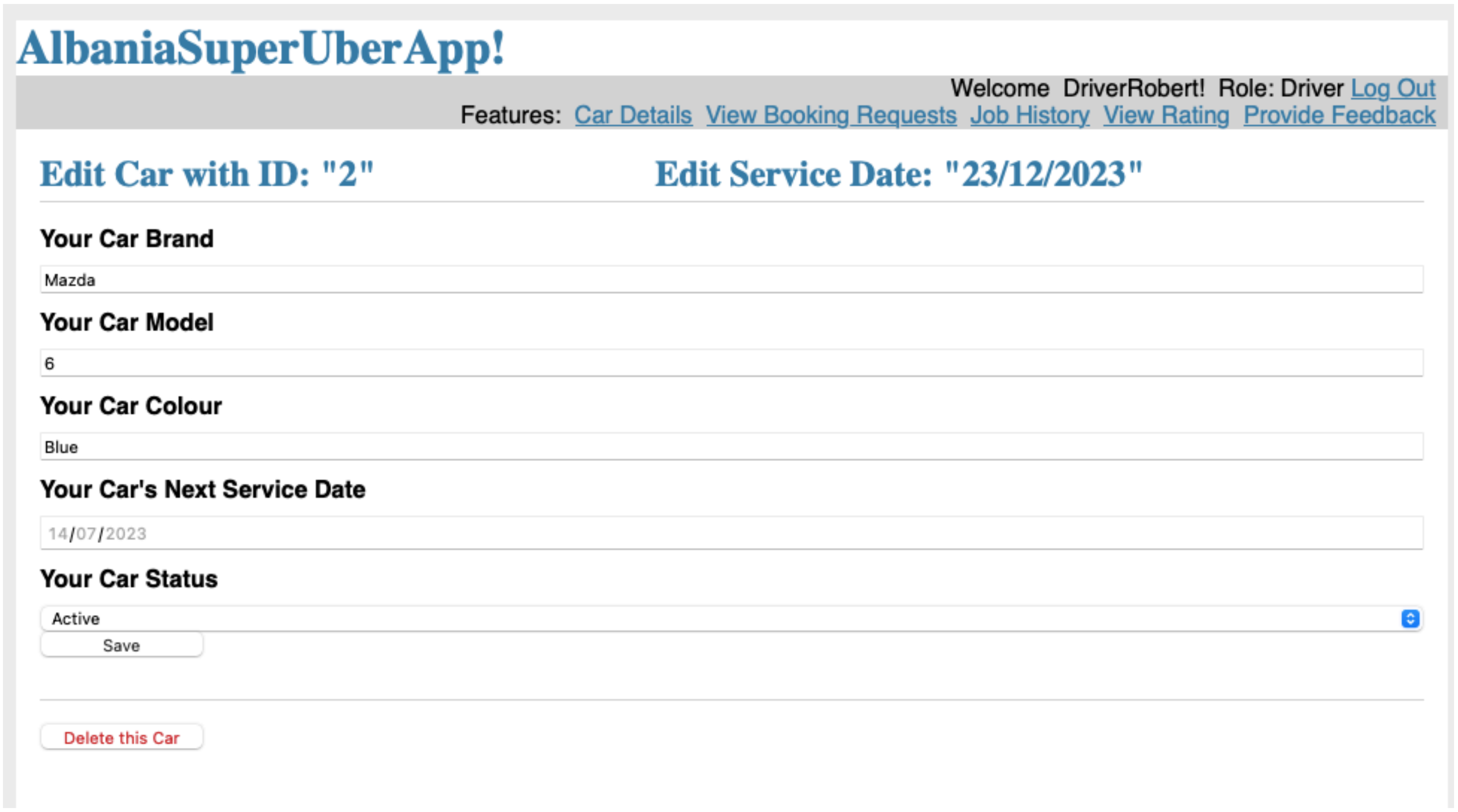
Thank you for the ride DriverRobert!

Describe/Tag your Feedback

Wembley-Euston

Figure 11. Screenshot showing the feedback feature

Description: This screenshot displays feedback already provided by the customer with the tag 'Wembley-Euston.' The screen shows the customer's feedback, allowing them to view and review their previous input. Customers can edit their feedback if they wish to modify their comments or provide additional details. Additionally, customers can choose to delete their feedback if they no longer want it to be associated with the ride or their account.


AlbaniaSuperUberApp!

Welcome DriverRobert! Role: Driver Log Out

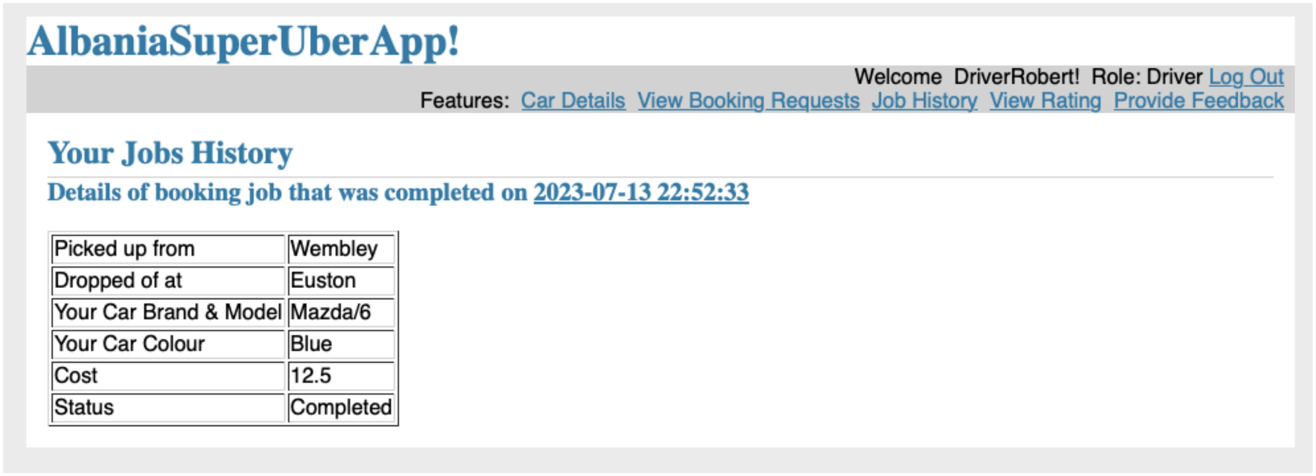
Features: Car Details View Booking Requests Job History View Rating Provide Feedback

Edit Car with ID: "2" Edit Service Date: "23/12/2023"

Your Car Brand	Mazda
Your Car Model	6
Your Car Colour	Blue
Your Car's Next Service Date	14/07/2023
Your Car Status	Active

Figure 12. Screenshot showing the driver's car details

Description: This feature includes essential information such as the car model, brand, status, color, and the car's next service date. These details provide visibility into the driver's vehicle and its current state. It allows the driver to manage their car information effectively within the app. Additionally, this feature provides the ability to edit the car details, enabling drivers to update information like the car model, brand, or color if there are any changes or updates.


AlbaniaSuperUberApp!

Welcome DriverRobert! Role: Driver Log Out

Features: Car Details View Booking Requests Job History View Rating Provide Feedback

Your Jobs History

Details of booking job that was completed on 2023-07-13 22:52:33

Picked up from	Wembley
Dropped of at	Euston
Your Car Brand & Model	Mazda/6
Your Car Colour	Blue
Cost	12.5
Status	Completed

Figure 13. Screenshot showing job history feature

Description: This screenshot showcases the job history feature for the driver, providing a comprehensive view of their past rides. The screen displays important information for each ride, such as the pick-up location, cost of the ride, car model used, and the status of the completed ride. It allows the driver to easily access and review their previous jobs, providing them with a record of their activity and earnings.

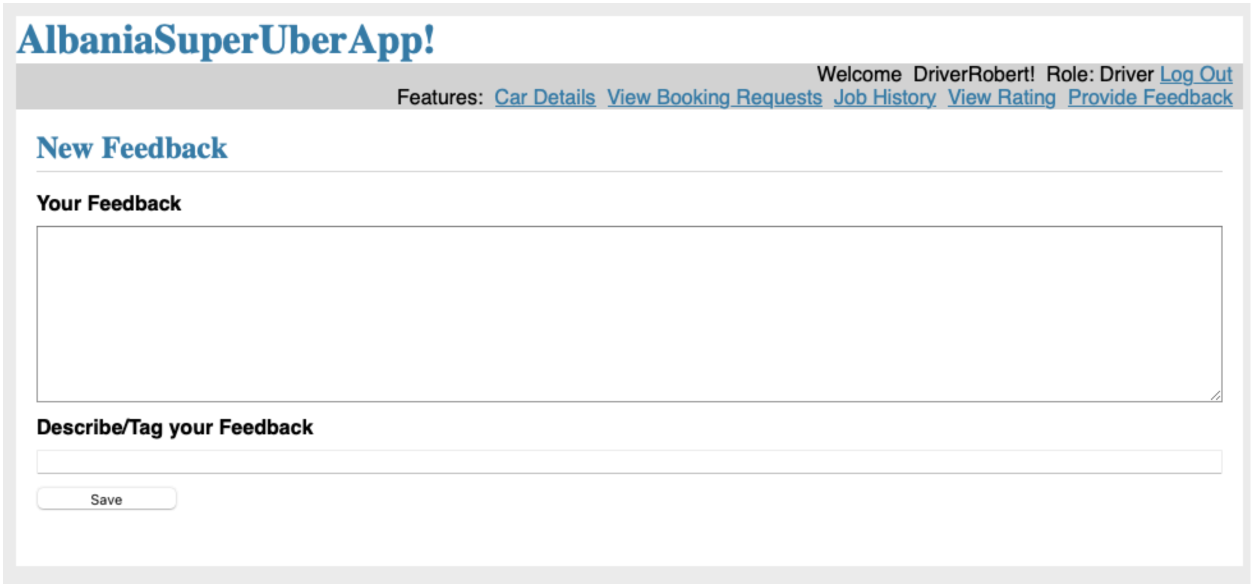
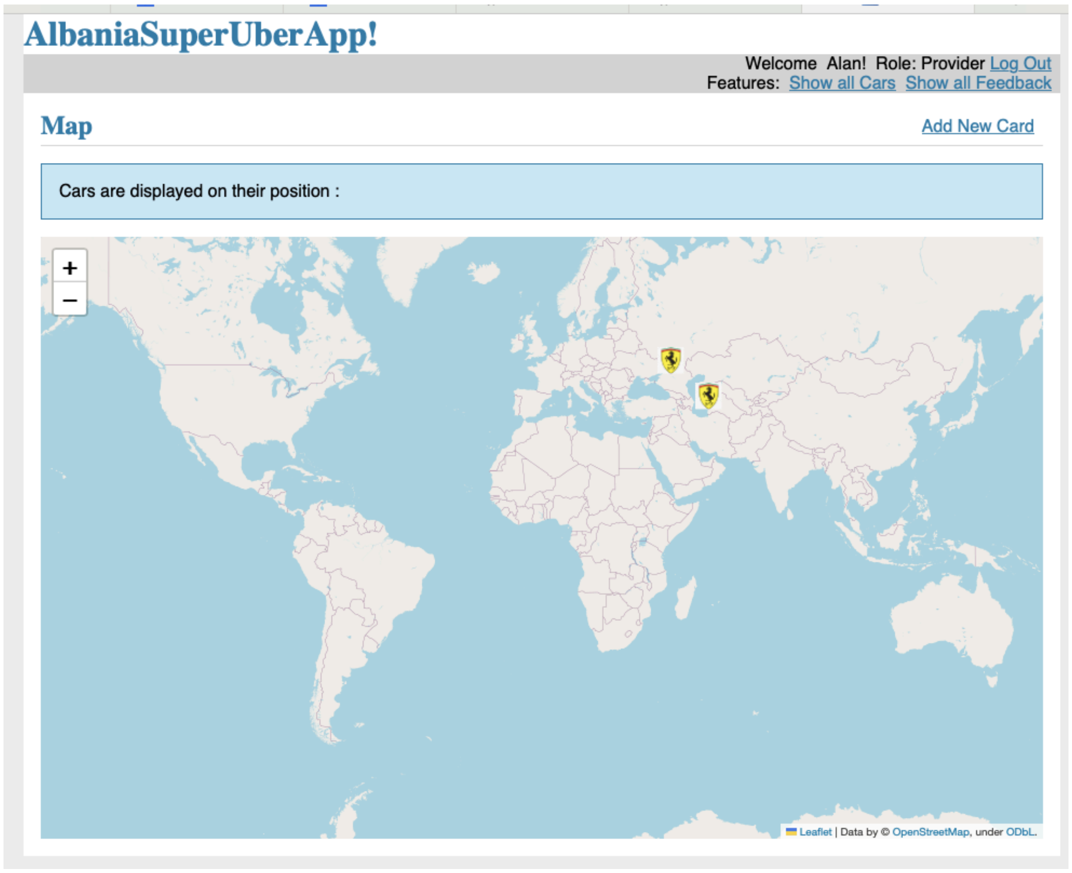

AlbaniaSuperUberApp!
Welcome DriverRobert! Role: Driver Log Out
Features: Car Details View Booking Requests Job History View Rating Provide Feedback
New Feedback
Your Feedback
Describe/Tag your Feedback
Save

Figure 14. Screenshot showing feedback feature for the driver

Description: This screenshot showcases the feedback feature for the drivers. After feedback is saved, drivers have the option to manage it further. They can choose to edit the feedback or to delete it.


AlbaniaSuperUberApp!

Welcome Alan! Role: Provider Log Out
Features: Show all Cars Show all Feedback

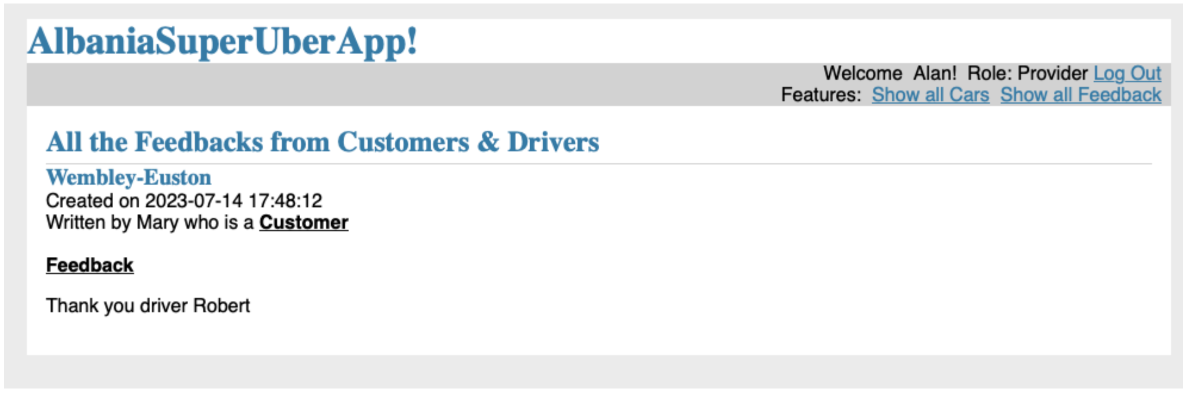
Map

Add New Card

Cars are displayed on their position :

Figure 15. Screenshot showing car positions

Description: The screen shows a map with the positions of cars in real-time. This feature allows the provider to monitor the locations of the cars.


AlbaniaSuperUberApp!

Welcome Alan! Role: Provider Log Out
Features: Show all Cars Show all Feedback

All the Feedbacks from Customers & Drivers

Wembley-Euston
Created on 2023-07-14 17:48:12
Written by Mary who is a **Customer**

Feedback

Thank you driver Robert

Figure 16. Screenshot showing feedback feature for the provider

Description: This feature allows the provider to have a centralized view of customer feedback, providing valuable insights into the overall satisfaction level and areas of improvement.

<u>UNIT TESTING:</u>

BOOKING	def test_show_cars_success(client):	Tests the successful retrieval of car information from the "showCars" API endpoint
	def test_bookcar(client):	Tests the functionality of the "bookcar" API endpoint
	def test_listRequests(client):	Tests the retrieval of customer requests
	def test_acceptJob(client):	Tests the successful acceptance of a job request
	def test_checkBooking(client):	Tests the functionality ofe“checkBooking".
	def test_startBooking(client):	Tests the functionality of starting a booking
	def test_completeBooking(client):	Tests the functionality of completing a booking
	def test_rate_driver(client):	Tests the functionality of rating a driver
CAR	def test_createCar(client):	Tests the functionality of creating a new car
	def test_updateCar(client):	Test the functionality of updating car details
	def test_deleteCar(client):	Test the functionality of deleting a car
	def test_getCar(client):	Tests the functionality of retrieving car information

CARD	def test_createCard(client):
	
	def test_listCard(client):
	
	def test_getCard(client):
	
	def test_updateCard(client):
	
	def test_deleteCard(client):
		Tests the functionality of creating a new payment card
		
		Tests the retrieval of payment cards
		
		Tests the functionality of retrieving card information
		
		Tests the functionality of updating a payment card
		
		Tests the functionality of deleting a payment card

FEEDBACK	def test_createFeedback(client):
	
	def test_listFeedback(client):
	
	def test_listAllFeedback(client):
	
	def test_getFeedback(client):
	
	def test_updateCard(client):
	
	def test_deleteFeedback(client):
		Tests the functionality of creating a new feedback
		
		Tests the retrieval of feedback for a specific user
		
		Tests the retrieval of all feedback
		
		Tests the retrieval of a specific feedback
		
		Tests the update of a specific feedback
		
		Tests the deletion of a specific feedback

JOB AND RATING	def test_listJob(client):
	
	def test_driverRating(client):
		Tests the retrieval of a list of job history
		
		Tests the retrieval of the driver rating

LINE OF CODES WRITTEN BY THE TEAM: 3549

FRONT-END	AUTH	login.html	22
		register.html	22
		auth.py	113
	BOOK	map.html	70
		book.py	312
	BOOKING	create.html	15
		listhistory.html	47
		listrequests.html	39
		update.html	20
		booking.py	74
	CAR	create.html	24
		list.html	29
		map.html	28
		update.html	32
		car.py	
	CARD	create.html	28
		index.html	38
		list.html	49
		update.html	34
		card.py	141
		db.py	52
	FEEDBACK	create.html	15
		Index.html	5
		list.html	30
		listAll.html	25
		update.html	19
		feedback.py	134
	JOB	job.html	47
		job.py	29
	RATING	driver_rating.html	56
		rating.py	47
	BASIC	base.html	42
		provider.py	56
		__init__.py	70

BACK-END	FLASKR	__inti__.py	77
		auth.py	96
		booking.py	230
		car.py	140
		card.py	141
		db.py	52
		feedback.py	154
		job.py	29
		rating.py	31
		schema.sql	89
	TESTS	test_booking.py	219
		test_car.py	129
		test_card.py	160
		test_feedback.py	178
		test_job_rating.py	60

Codebases:

We utilized existing codebases as a foundation for our project. Specifically, we continued building upon the tutorial provided by Flask Pallet Projects, which can be found at the following URLs:

Main Flask Pallet Projects documentation: https://flask.palletsprojects.com

Flask tutorial: https://flask.palletsprojects.com/en/2.3.x/tutorial/

These resources provided us with valuable guidance and examples, allowing us to leverage the Flask framework effectively in our project.

6. **Project Management**

Software Methodology

To run the entire project end to end, we depended on the industry standard and trustworthy agile methodology. We utilized the Scrum approach for the entire project. Some of us in the team have had some experience leading/being involved in a Scrum approach for projects before so it became much easier to set it as a standard. Moreover, it was also taught intensively in a previous module called Information Systems. Our "Kanban Board" was setup in an online tool called "Kanban Flow". Prior to selecting that, we researched and trialed out various tools including one recommended by Keith called "Trello". However, we found the latter an overkill as we already had MS Teams set up for the standard meetings and a WhatsApp group for informal communication. We were very cautious into finding simple and streamlined tools that made the entire project management conducive and efficient, rather than an overload of tools that incorporated buzzwords (less is more mantra).

Tech Tools Utilized

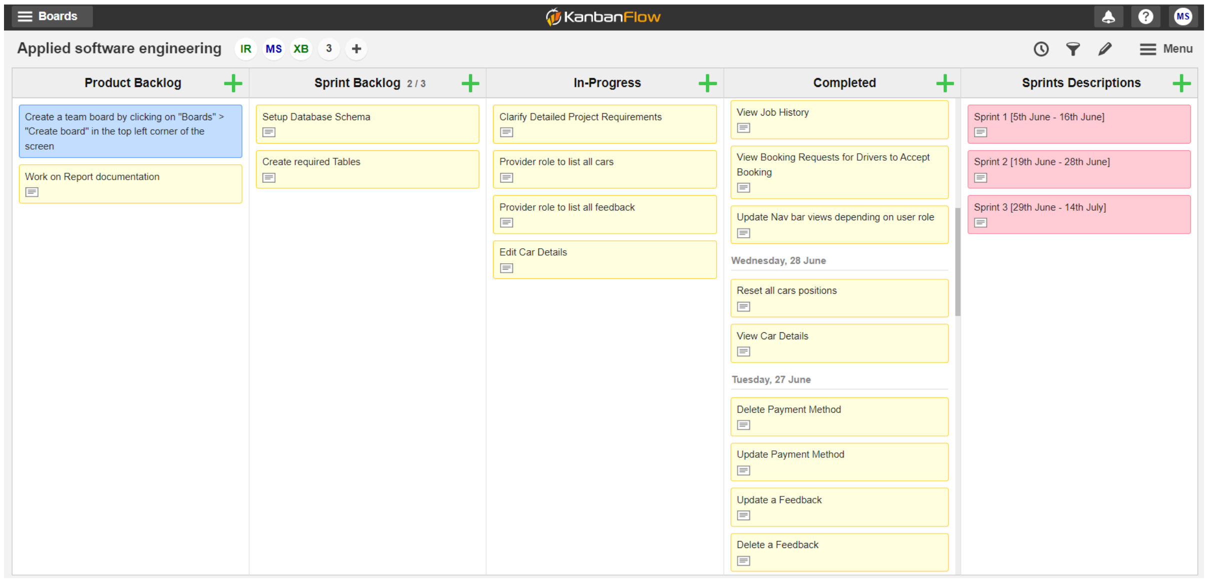
Use-Case	Tech Tool
To Host/Aid Meetings Online	MS Teams
To be our Kanban Board to hold Product Backlog & More	Kanban Flow Our team's board link: *Applied software engineering - KanbanFlow*

To Host our shared repositories	Frontend repository (it is public):
	melvir86/ase: This is a shared code repository for Group 3's Applied Software Engineering (ASE) project (github.com)
	Backend repository (it is public):
	melvir86/ase-backend: Backend for our ASE Ride Hailing project (github.com)

To be our individual application coding/ testing platforms	Codio (Melvir):
	Codio - Teach Code. With confidence.
	Codio (Xhulio):
	Codio - Teach Code. With confidence.
	Codio (Irdi):
	Codio - Teach Code. With confidence.

Significant Events

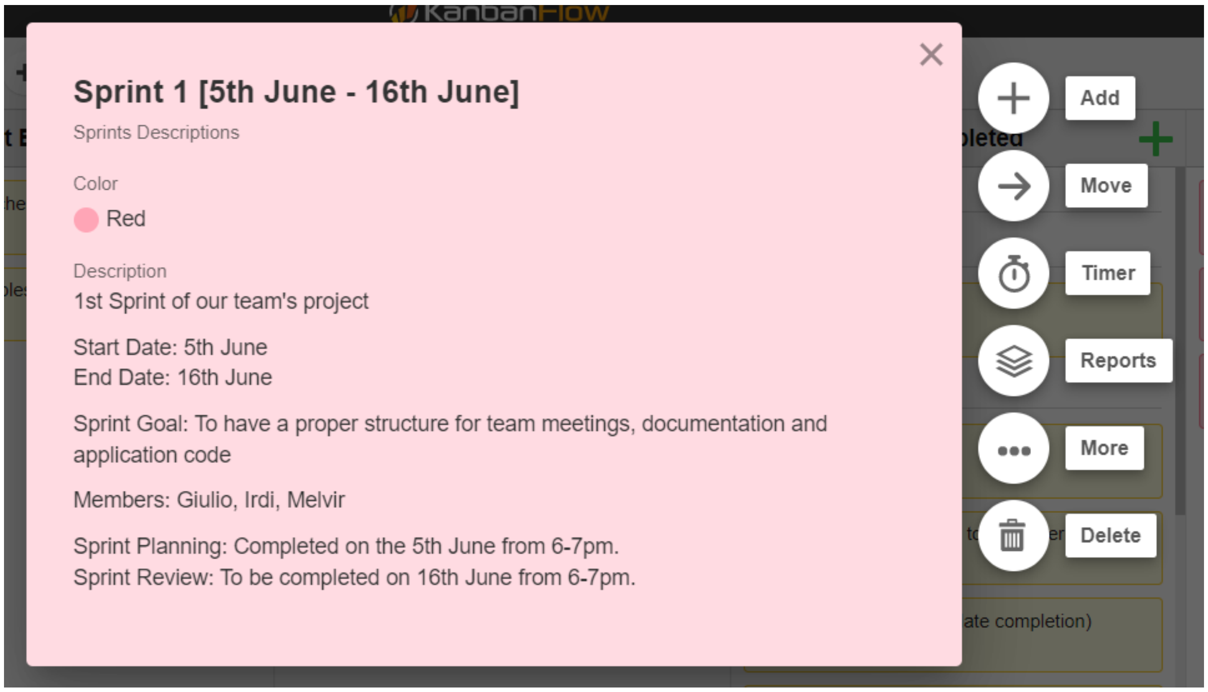
Naturally the most significant events encompass around the 3 Sprints we ran from start to end. We have copied the material here from our Kanban Board. Prior to explaining the 3 Sprints, we have attached a screenshot showing details of our Kanban Board.


Boards
KanbanFlow
Applied software engineering IR MS XB 3 + Menu

Product Backlog	Sprint Backlog 2 / 3	In-Progress	Completed	Sprints Descriptions
Create a team board by clicking on "Boards" > "Create board" in the top left corner of the screen	Setup Database Schema	Clarify Detailed Project Requirements	View Job History	Sprint 1 [5th June - 16th June]
Work on Report documentation	Create required Tables	Provider role to list all cars	View Booking Requests for Drivers to Accept Booking	Sprint 2 [19th June - 28th June]
		Provider role to list all feedback	Update Nav bar views depending on user role	Sprint 3 [29th June - 14th July]
		Edit Car Details	Wednesday, 28 June	
			Reset all cars positions	
			View Car Details	
			Tuesday, 27 June	
			Delete Payment Method	
			Update Payment Method	
			Update a Feedback	
			Delete a Feedback	

<u>Sprint 1</u>

<u>Description</u>



Sprint 1 [5th June - 16th June]

Sprints Descriptions

Color

● Red

Description

1st Sprint of our team's project

Start Date: 5th June
End Date: 16th June

Sprint Goal: To have a proper structure for team meetings, documentation and application code

Members: Giulio, Irdi, Melvir

Sprint Planning: Completed on the 5th June from 6-7pm.
Sprint Review: To be completed on 16th June from 6-7pm.

Product Backlog Items (PBIs) worked on

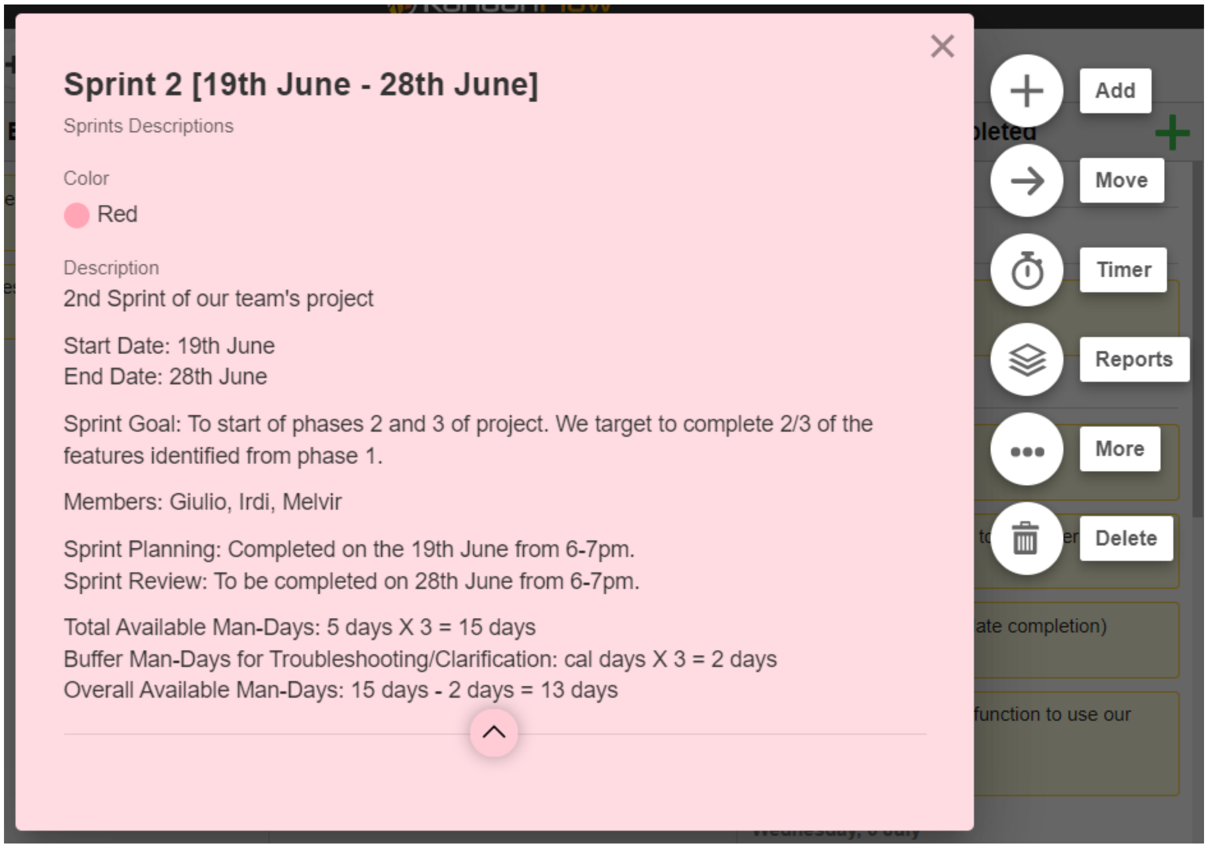
Number	Item
1	Setup shared code repository
2	Research Shared platform for Product Management
3	Come up with Project Documentation Outline
4	Come up with Documentation for Scenarios
5	Come up with Documentation for Personas
6	Create Car class to store provided sample data to persist in database

Major meetings/discussions

Description	Date	Recording (if any)
1. Sprint 1 Planning	9th June 2023	Unfortunately the recording had an issue, thus we could not get a link from MS Teams
2. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days]	12th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/EV7tJSKXIOBNtejz4o7kxQ4BqwOXvG3OCZmy-zX9KpEXJw
3. 3. Sprint 1 Review	16th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/Ee2avMIPjWtMj0lwl8ZOANIBk41WuE9U8WauDh_QjFqohw

Sprint 2

Description



Sprint 2 [19th June - 28th June]

Sprints Descriptions

Color

● Red

Description

2nd Sprint of our team's project

Start Date: 19th June
End Date: 28th June

Sprint Goal: To start of phases 2 and 3 of project. We target to complete 2/3 of the features identified from phase 1.

Members: Giulio, Irdi, Melvir

Sprint Planning: Completed on the 19th June from 6-7pm.
Sprint Review: To be completed on 28th June from 6-7pm.

Total Available Man-Days: 5 days X 3 = 15 days
Buffer Man-Days for Troubleshooting/Clarification: cal days X 3 = 2 days
Overall Available Man-Days: 15 days - 2 days = 13 days

Product Backlog Items (PBIs) worked on

Number	Item
1	Come up with Documentation for User Stories
2	Come up with Documentation for Feature Identification
3	User Registration
4	User Login
5	Display Car positions on Map
6	Make a Booking
7	Cancel a Booking
8	List all Payment Methods

9	Create new Payment Method
10	Update Payment Method
11	Delete Payment Method
12	Reset all Cars Positions

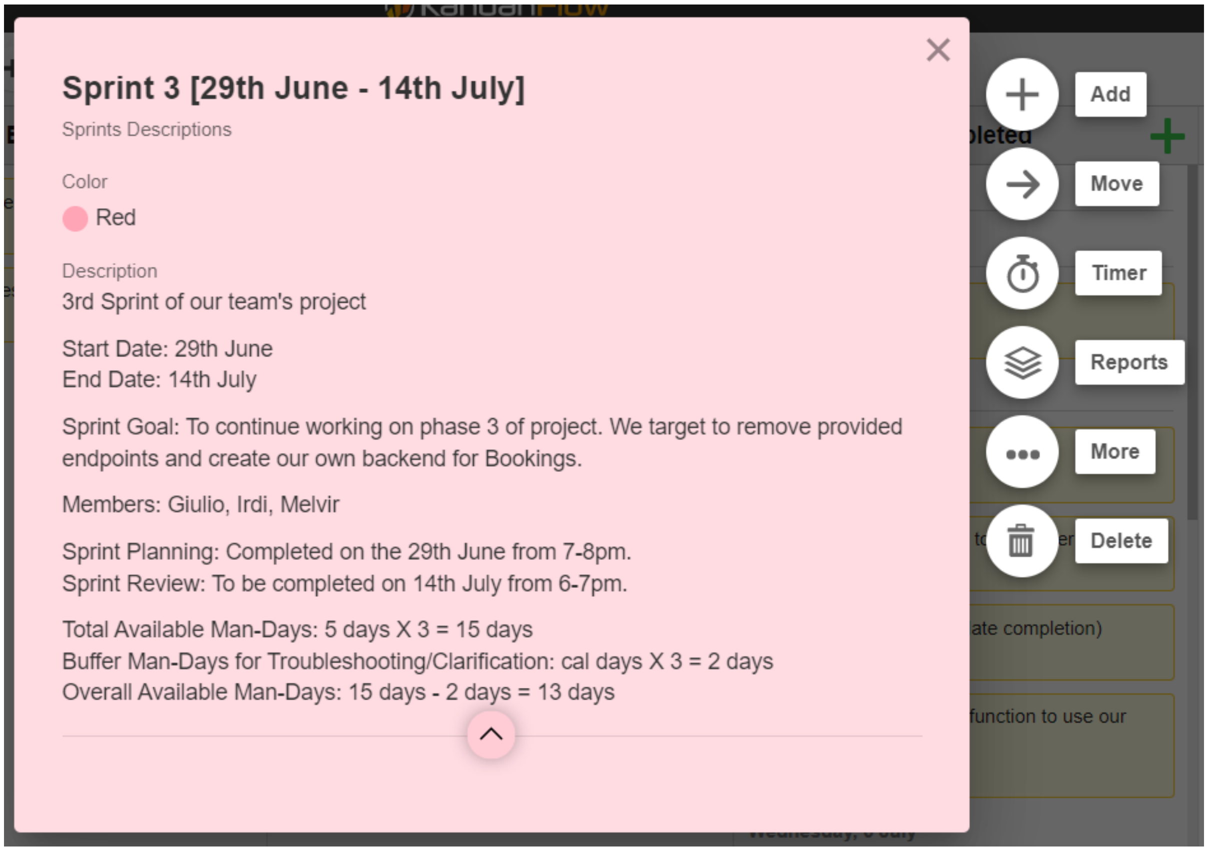
<u>Major meetings/discussions</u>

Description	Date	Recording (if any)
4. Sprint 2 Planning	19th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/ETlqbuultI5FgM8alj8Y2IQBSQn2GyZOvNlmKImP9LLysA
5. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days]	24th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/ESE5g1Q7oLpNufML8KgzREIBurCeOSEoWjYhQUpgAxOImQ
6. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days]	28th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/EVVg_RNaWaFHkeuktROAtScBFoOcpyikgQtSK4INfj5gAA

7. Sprint 2 Review	28th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/ETKz_BMB1bBGvWTU75NzIw8BA1kXU5yA7x1Ui28cgMursw

Sprint 3

Description


Sprint 3 [29th June - 14th July]

Sprints Descriptions

Color

● Red

Description

3rd Sprint of our team's project

Start Date: 29th June
End Date: 14th July

Sprint Goal: To continue working on phase 3 of project. We target to remove provided endpoints and create our own backend for Bookings.

Members: Giulio, Irdi, Melvir

Sprint Planning: Completed on the 29th June from 7-8pm.
Sprint Review: To be completed on 14th July from 6-7pm.

Total Available Man-Days: 5 days X 3 = 15 days
Buffer Man-Days for Troubleshooting/Clarification: cal days X 3 = 2 days
Overall Available Man-Days: 15 days - 2 days = 13 days

Product Backlog Items (PBIs) worked on

Number	Item
1	Clarify Detailed Project Requirements
2	Simulate Car movement to Customer

3	Change architecture from monolith to Client & Server architecture
4	Update Book Car function to use our own logic
5	Design Database Schema
6	Create Required Tables
7	View Booking Requests for Drivers to Accept Booking
8	Simulate Button (to simulate Trip Completion)
9	Submit Driver Rating
10	View Driver Rating
11	View Job History (for Drivers)
12	View Booking History (for Customers)
13	Update Navigation bar views depending on user role upon login
14	Provider role to list all cars
15	Provider role to list all feedback

<u>Major meetings/discussions</u>

Description	Date	Recording (if any)
Sprint 3 Planning	28th June 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/ETI7cT8uJhxGosLitJk7SKYBtEqinVwhBHFus66nj1H7fA

Daily Standup Simulation [what we did over the last 7 days and what we plan on working on for the next 7 days]	5th July 2023	(Part 1) - https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/EfCCz7Hq9AJJoyiEUM9P068BzNNwQKyBca1s6FFODKoBJw (Part 2) - https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/Efc0fF_PuOFCnwBNekLPkw0BlRYk-rHUjSU9TAT7HjhB7A
Daily Standup Simulation [what we did over the last 7 days and what we plan on working on for the next 7 days]	12th July 2023	https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/Ef3nTa7sQ01JmmAVUdv7XpgBSjOYUlEs6_gtDBXnL4fatQ
Sprint 3 Review	14th July 2023	Unfortunately the recording had an issue, thus we could not get a link from MS Teams

Code Structure (Github repository that we pulled for our Codio platform)

We have 2 applications to serve the product – ase (frontend) & ase-backend (backend). There are multiple branches for each of this application that is described in the table below.

Application	Branches	Rationale

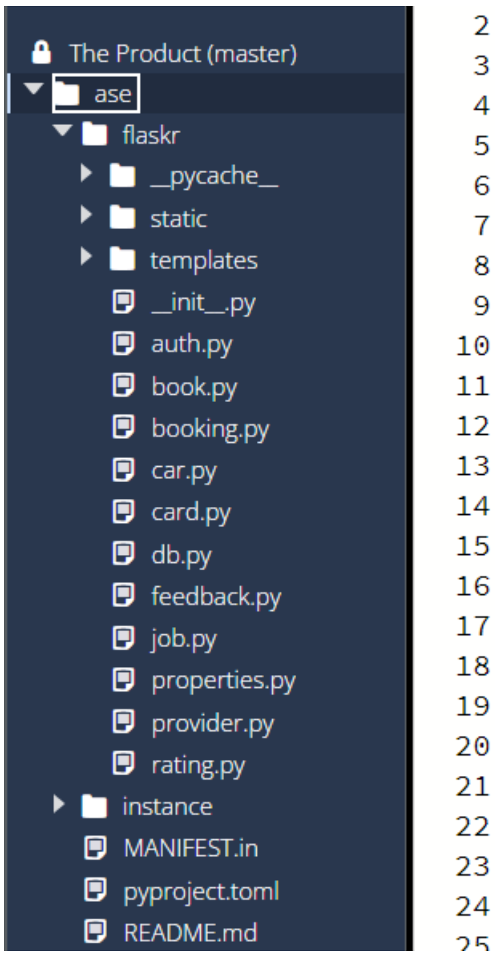
ase	main	Holds documentation folder (scenarios, stories, personas, feature-identifications, tech diagrams, project report) and meeting recording links
	phase-one	Holds project code for phase-one functionalities and README file (to specify how to start the application)
	phase-two	Holds project code for phase-two functionalities and README file (to specify how to start the application)
	phase-three	Holds project code for phase- three functionalities and README file (to specify how to start the application)
ase-backend	main	Nil
	phase-three	Holds project code for phase-three backend functionalities and README file (to specify how to start the application)

Project Structure

Frontend

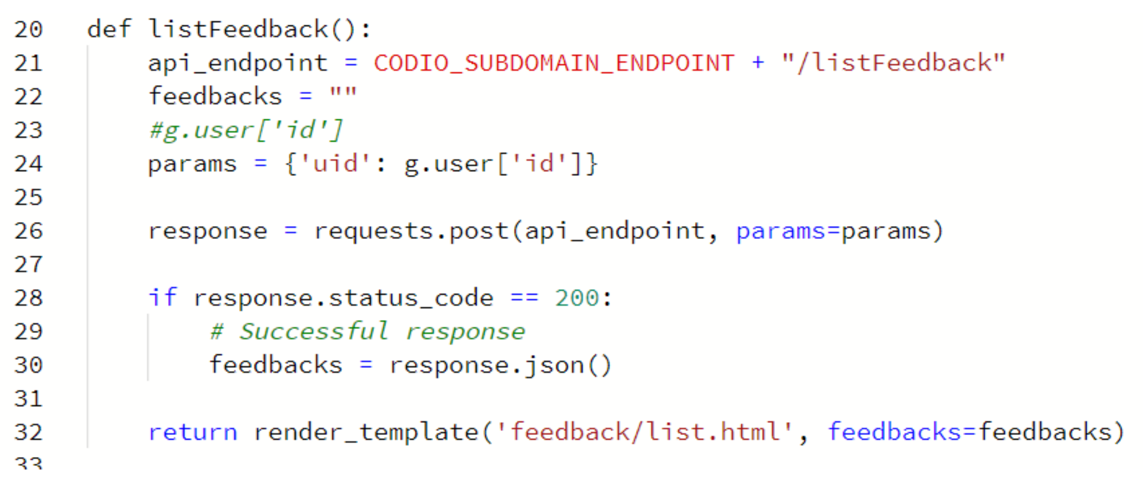
For the frontend, our team decided to have just 2 subfolders (static & templates). The former holds the CSS file while the latter holds the various HTML templates to be served to the client (user).

We also have individual component py files that serve as the middleman between the frontend and backend applications. These py files (1) retrieves information from the HTML templates and calls the backend APIs with the required payloads & (2) sends the information to be loaded to the HTML templates using the JSON results returned from the backend APIs.



```
The Product (master)
  ase
    flaskr
      __pycache__
      static
      templates
      __init__.py
      auth.py
      book.py
      booking.py
      car.py
      card.py
      db.py
      feedback.py
      job.py
      properties.py
      provider.py
      rating.py
    instance
    MANIFEST.in
    pyproject.toml
    README.md
```

Below is an example of the Feedback component doing what was described above. In this scenario, the Feeback components calls the backend API endpoint and returns the resulting JSON to the Feedback template (feeback/list.html).

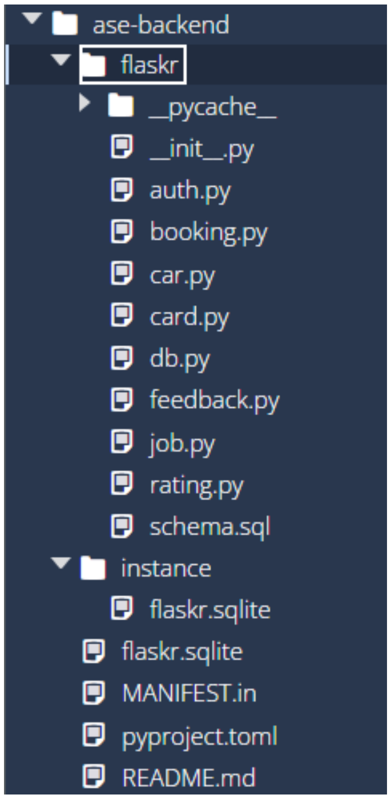


```python
def listFeedback():
    api_endpoint = CODIO_SUBDOMAIN_ENDPOINT + "/listFeedback"
    feedbacks = ""
    #g.user['id']
    params = {'uid': g.user['id']}

    response = requests.post(api_endpoint, params=params)

    if response.status_code == 200:
        # Successful response
        feedbacks = response.json()

    return render_template('feedback/list.html', feedbacks=feedbacks)
```

<u>Backend</u>

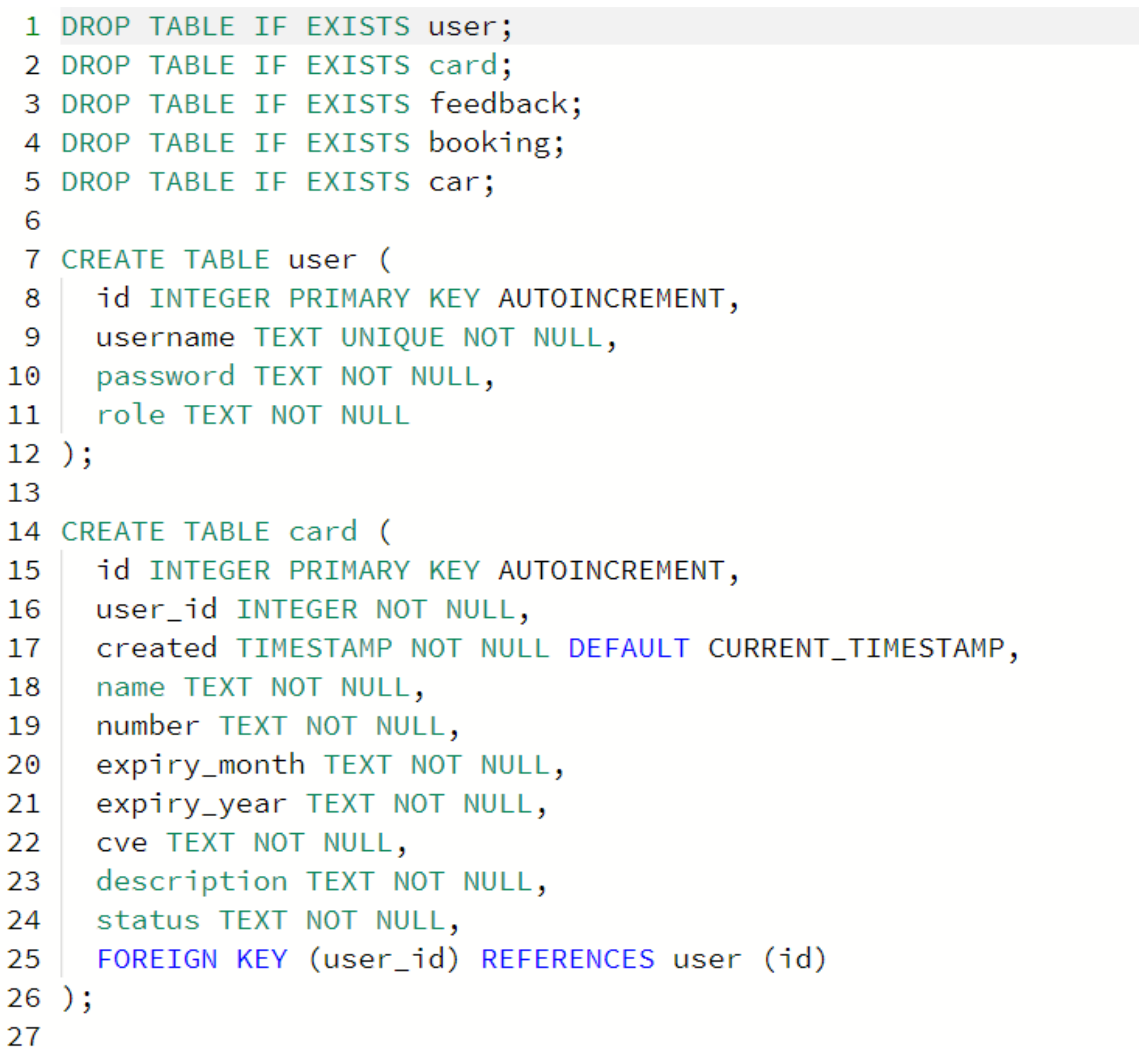
For the backend, it is more straightforward since its responsibility is only to serve API routes. Thus we just segregated the API files (py files) according to their different business domains (ie. car.py, feedback.py, etc).

Since the backend requires a database (SQLite) to store/retrieve data from, we have sqlite file that is stored in instance/flaskr.sqlite when initialized (flask --app flaskr init-db).

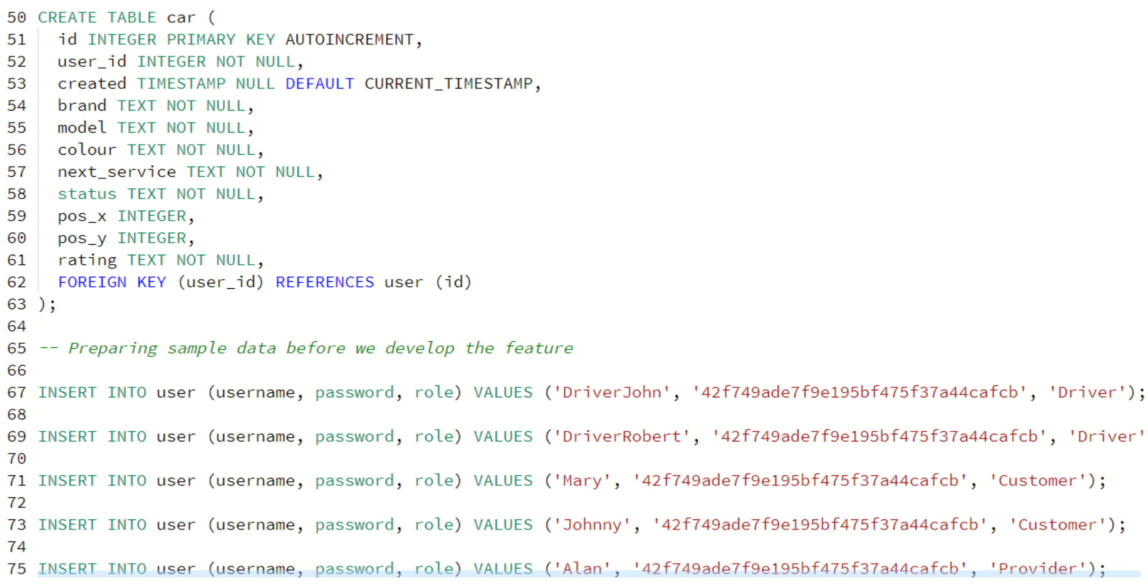


```
ase-backend
  flaskr
    __pycache__
    __init__.py
    auth.py
    booking.py
    car.py
    card.py
    db.py
    feedback.py
    job.py
    rating.py
    schema.sql
  instance
    flaskr.sqlite
  flaskr.sqlite
  MANIFEST.in
  pyproject.toml
  README.md
```

We also have a schema.sql that contains all the tables creation as well as for sample data to be added as shown in the screenshot, which allows us to test the application easily.

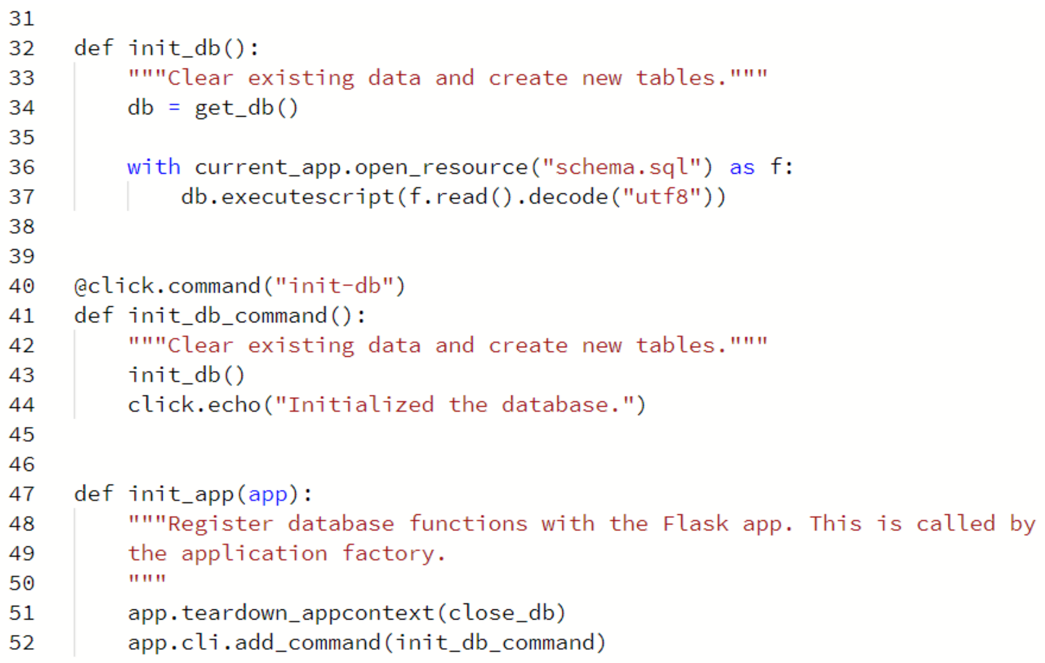


```sql
DROP TABLE IF EXISTS user;
DROP TABLE IF EXISTS card;
DROP TABLE IF EXISTS feedback;
DROP TABLE IF EXISTS booking;
DROP TABLE IF EXISTS car;

CREATE TABLE user (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL,
  role TEXT NOT NULL
);

CREATE TABLE card (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  name TEXT NOT NULL,
  number TEXT NOT NULL,
  expiry_month TEXT NOT NULL,
  expiry_year TEXT NOT NULL,
  cve TEXT NOT NULL,
  description TEXT NOT NULL,
  status TEXT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES user (id)
);

```



```sql
CREATE TABLE car (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  created TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  brand TEXT NOT NULL,
  model TEXT NOT NULL,
  colour TEXT NOT NULL,
  next_service TEXT NOT NULL,
  status TEXT NOT NULL,
  pos_x INTEGER,
  pos_y INTEGER,
  rating TEXT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES user (id)
);

-- Preparing sample data before we develop the feature

INSERT INTO user (username, password, role) VALUES ('DriverJohn', '42f749ade7f9e195bf475f37a44cafcb', 'Driver');

INSERT INTO user (username, password, role) VALUES ('DriverRobert', '42f749ade7f9e195bf475f37a44cafcb', 'Driver');

INSERT INTO user (username, password, role) VALUES ('Mary', '42f749ade7f9e195bf475f37a44cafcb', 'Customer');

INSERT INTO user (username, password, role) VALUES ('Johnny', '42f749ade7f9e195bf475f37a44cafcb', 'Customer');

INSERT INTO user (username, password, role) VALUES ('Alan', '42f749ade7f9e195bf475f37a44cafcb', 'Provider');
```

Lastly, we also have a single db.py that initializes the DB using the schema.sql file. This file also allows the various backend component python files to retrieve a DB connection easily, and in a standardized manner. This avoids us having to duplicate code everywhere, resulting in good code quality and reusability.



```python

def init_db():
    """Clear existing data and create new tables."""
    db = get_db()

    with current_app.open_resource("schema.sql") as f:
        db.executescript(f.read().decode("utf8"))


@click.command("init-db")
def init_db_command():
    """Clear existing data and create new tables."""
    init_db()
    click.echo("Initialized the database.")


def init_app(app):
    """Register database functions with the Flask app. This is called by
    the application factory.
    """
    app.teardown_appcontext(close_db)
    app.cli.add_command(init_db_command)
```

7. Team

Backgrounds

As shown in the table below, the only person with experience / expertise of building such applications/products was Melvir. Thus, it was a very uphill task from start of project to the end for the entire team. The team had to put up many late nights so that we were able to deliver a good product with solid code standards on time.

Team Member	Experience of IT / Tech	Experience building such a product	Experience with Scrum & Scrum tools

Melvir Singh	5 years as Software Engineer, 3 years in DevOps and 2 years as Product Manager	Vast experience in building frontend and backend applications with specialization of containerized microservices	Vast experience in leading teams with Scrum methodology
Dalird Bufi	3 years as Infrastructure Engineer	Nil	Nil
Xhulio Bishtaja	Completely new to IT / Tech	Nil	Nil

We strongly believe that the team is a very well-functioning one, and one that allowed all team members to put in their fair share of work. With Melvir leading the team, the rest were able to pick up on the code standards and come up with features completely on their own later on. This further reiterates the importance of spending a fair bit of time laying good foundation that everyone in the team can appreciate and learn from.

Team Member	Role	Features Contributed	Rough Percentage Contributions
Melvir Singh	Lead Architect / Developer	Setting Code Standards & Best Practice Customer – List all Payment Cards Customer – Submitting new Payment Card Customer – Updating a Payment Card Customer – Deleting a Payment Card Customer – View Booking History Driver – View Job History Driver – Accept Booking Requests Provider – Show all Feedback Final Report	33.33%
Dalird Bufi	Core Developer	Customer – Booking Car Customer – Simulate Starting of Ride Customer – Simulate Completion of Ride Driver – List all car details Driver – Submit new car details Provider – Show all Cars on Map Unit Testing on all Backend components Creating Test/Sample Data for Testing Final Report	33.33%

Xhulio Bishtaja	Core Developer	Customer – List all Feedback Customer – Submitting new Feedback Customer – Updating a Feedback Customer – Deleting a Feedback Customer – Submit Driver Rating Driver – View Driver Rating Driver – Updating car details Driver – Deleting car details Announcement Page Final Report	33.33%

8. Constraints and Risks

CONSTRAINTS	There were no significant constraints with social, ethical, policy, or legal difficulties throughout the development of the "AlbaniaSuperUberApp" ride-hailing service. The group took proactive measures to guarantee user security, preserve user privacy, and achieve fairness. Overall, the development process progressed without any major issues in this area.

RESOURCES	The project's development was significantly facilitated by class tutorials and the Codio platform, providing comprehensive guidance and support. Exploring online tutorials on Flask and Pallets projects further expanded knowledge and skills, ultimately increasing the learning experience and ensuring the successful completion of the project.
SUGGESTIONS	Would have been useful to have additional tutorials or classes into more advanced features or functions we could have utilised on python or flask.

9. Reflection

Lessons Learnt

As a team, we feel there are a couple of lessons we have all learnt.

Firstly, that it is imperative to pick up knowledge of technologies and new syntax using online tutorials as what is covered in the short duration of a Masters' module class would not be sufficient enough to be able to come up with good solutions for the project. What was taught in class was indeed useful to us but it only covered the basics whilst we required a whole lot more advanced functions to complete the features we wanted to for our product.

Secondly, being able to adapt to each other's working styles early on would indeed have been more useful. As it was the first time we were working on a project together, we did not know or understand each other's working style as well as availabilities in terms of schedules. As expected, during the initial team storming stage, we encountered many arguments and misunderstandings which took time to clear up. Thus, our lesson for this was that it would be helpful to have a get to know session outside of work to form that bond and common expectation prior.

What Went Well & What Did Not

What Went Well	What Did Not Go So Well
Managed to notch up a really good and effective working relationship towards the end of the project (the team concurrently worked through Unit Testing via Dalird and Report via Melvir & Xhulio) through a really tight timeline	Had many arguments at the start of the project primarily because we did not really know each other's styles
Developed successfully all the core features we wanted to deliver for Customer, Driver and Provider role	Wanted to get a further edge over other teams by delivering Advanced Features such as AI that could automatically recommend to a Driver where to position his/her car but unfortunately we could not find sufficient time to pick up the additional skills
Github & Kanban Flow were really good tools for sharing code repository and managing Sprint cycles / tasks respectively	Codio for developing our code was unfortunately disappointing. It constantly crashed resulting in us losing some key commands and having to restart the platform a couple of times (resulting in a waste of time)

Recommendations

For future teams that embark on such a module / project, we have these 3 advices. These advices are also some that we would have done differently ourselves if we could restart the project.

Firstly, as mentioned before, not to wait for the project to begin but to start doing tutorials available online to pick up the basic/core technologies required much earlier for Flask, Python, HTML, CSS, etc. If we had done this, by the time the project requirements came out, we would have hit the ground running and would have had much more time for the advanced AI features that we wanted for the additional “wow” factor.

Secondly, to have a good outside “work” relationship with the other team members. This can be done via a simple dinner / drinks meet. The benefit of doing this is to form a strong bond that will ensure that everyone can be honest with each other and be upfront over what they aim to achieve for the project.

Lastly, to request for additional help sessions from Keith as we realized towards the end of the project that some other teams did not have the basic knowledge. That is really detrimental to the project and would result in not being able to deliver the most basics of features. Thankfully, we were always alert during lessons and kept asking questions/clarifications with Keith/Steven whenever we were unsure about certain tasks/technologies/libraries.